

Practical Microcontroller Programming with ARDUINO

เรียนง่ายเป็นเร็ว



เปลี่ยน Arduino ให้เป็นเรื่องง่าย ให้ทุกคนเรียนได้อย่างรวดเร็ว

หนังสือเรียนไมโครคอนโทรลเลอร์ และการเขียนโปรแกรมควบคุมการทำงานด้วย Arduino เหมาะสำหรับผู้เริ่มต้นที่ไม่มีพื้นฐานมาก่อน

ผู้แต่ง ทศพล บ้านคลองสี่
อาจารย์มหาวิทยาลัยและวิทยากรพิเศษทั้งภาครัฐและเอกชน
เจ้าของเพจ/ยูทูบ : Ai, NesT The Series และ GlurGeek.com

Source Code
SERAZU.COM



มีเพียง “**ความรู้**” เท่านั้นที่มนุษย์ใช้พลิก “**โลก**”
และเปลี่ยนชีวิต เราจึงสร้างสรรค์ และส่งมอบ “**ความรู้**”
ในรูปแบบที่ดีกว่า เพื่อให้คนไทย “**เรียนรู้**” ได้ตลอดชีวิต

Only “**Knowledge**” can help human
change “**The World**” and “**Their Lives**”.
With this truth, it drives us to deliver
“**Knowledge**” for Thai being able to
“**Learn**” better everyday.











Practical Microcontroller & Programming with ARDUINO
เรียนง่ายเป็นเร็ว

Writer	ทศพล บ้านคลองสี่
Editor	ภีรพล คชาเจริญ
Graphic Designer	ชวนันท์ รัตนะ
Page Layout	สุรัสวดี วงศ์จันทร์สุข
Proofreader	สุนทรี บรรลือศักดิ์
Publishing Coordinators	วรพล ณธิกุล, สุพัตรา อาจปรุ, ศรัณย์ คมขำ

สร้างสรรค์โดย



พิมพ์ครั้งที่ 1 พฤษภาคม 2565

ข้อมูลทางบรรณานุกรม

ทศพล บ้านคลองสี่
Practical Microcontroller & Programming with ARDUINO
เรียนง่ายเป็นเร็ว
นนทบุรี : ไอดีซีฯ, 2565
336 หน้า
1. การเขียนโปรแกรมโดยใช้ภาษาโปรแกรมเฉพาะชนิด
I ชื่อเรื่อง
005.262
ISBN 978-616-487-315-5
ราคา 450 บาท

จัดพิมพ์และจัดจำหน่ายโดย



บริษัท ไอดีซี พรีเมียร์ จำกัด
200 หมู่ 4 ชั้น 19 ห้อง 1901
อาคารจัสมินอินเตอร์เนชั่นแนลทาวเวอร์
ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120
โทรศัพท์ 0-2962-1081 (อัตโนมัติ 10 คู่สาย)
โทรสาร 0-2962-1084

สมาชิกสัมพันธ์
โทรศัพท์ 0-2962-1081-3 ต่อ 121
โทรสาร 0-2962-1084

ร้านค้าและตัวแทนจำหน่าย
โทรศัพท์ 0-2962-1081-3 ต่อ 112-114
โทรสาร 0-2962-1084



PREFACE

นับตั้งแต่ที่ Arduino เปิดตัวมา ก็ได้กลายมาเป็นหนึ่งในโอเพ่นซอร์สฮาร์ดแวร์ที่ประสบความสำเร็จอย่างมากในวงการการสร้างนวัตกรรมสิ่งประดิษฐ์ ผมได้ซื้อ Arduino ตัวแรกมาเมื่อหลายปีก่อน เพื่อนำมาใช้สอนให้กับนักศึกษาและผู้ที่สนใจด้วยความอยากรู้อยากเห็นว่า อุปกรณ์ตัวนี้มีความน่าสนใจอย่างไร ทำไมมีแต่คนพูดถึงกัน จึงได้เริ่มศึกษาค้นคว้าอย่างจริงจังจนได้มีโอกาสดูคลิปของ Massimo Banzi ผู้คิดค้น Arduino ที่ขึ้นพูดบนเวทีในงาน TED Talk ทำให้ผมรู้สึกประทับใจในเจตนารมณ์ของพวกเขาที่ว่า พวกเขากำลังพยายามสร้างไมโครคอนโทรลเลอร์ที่มีขนาดกะทัดรัด ราคาไม่แพง ใช้งานง่าย สามารถเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์พื้นฐาน และอุปกรณ์เซนเซอร์แบบต่างๆ ได้

โปรแกรมที่ใช้ในการเขียนโปรแกรมสำหรับสั่งงานบอร์ด Arduino ก็สามารถดาวน์โหลดได้ฟรี พร้อมทั้งรองรับการเขียนโปรแกรมภาษาซี ซึ่งเป็นภาษาพื้นฐานที่เหมาะกับผู้เริ่มต้นในการพัฒนาโปรแกรมทางด้านฮาร์ดแวร์ที่เรียนรู้ได้ง่าย เราสามารถหาซื้อบอร์ดนี้ได้ทั่วไป อีกทั้งยังมีแหล่งข้อมูลให้ศึกษา และสอนวิธีการใช้งานมากมายบนโลกออนไลน์ ทั้งจากเว็บ arduino.cc ที่เป็นของผู้พัฒนาเอง และแหล่งชุมชนต่างๆ ซึ่งเป็นสิ่งที่ดีมากสำหรับผู้เริ่มต้นที่อยากจะเรียนรู้การเขียนโปรแกรมร่วมกับฮาร์ดแวร์ เพื่อสร้างนวัตกรรมสิ่งประดิษฐ์ได้ด้วยตัวเอง

Arduino จึงเป็นบอร์ดที่จะช่วยสร้างแรงบันดาลใจใหม่ๆ ให้กับคนธรรมดาอย่างเราๆ ทั้งเด็กและผู้ใหญ่ ให้สามารถกล้าคิดค้น กล้าใส่ความคิดสร้างสรรค์ในการออกแบบ ส่งเสริมการพัฒนาทักษะการเขียนโปรแกรม เพื่อสร้างนวัตกรรมสิ่งประดิษฐ์ใหม่ๆ แบบง่ายๆ ได้ด้วยตัวเอง ตั้งแต่งานศิลปะเชิงโต้ตอบไปจนถึงการสร้างหุ่นยนต์ก็สามารถทำได้จริง

ดังนั้น หนังสือเล่มนี้ได้เรียบเรียงเนื้อหามาเพื่อสร้างแรงบันดาลใจ และประโยชน์ให้กับผู้เริ่มต้นพัฒนาโปรแกรมในการจะก้าวเข้าสู่นักประดิษฐ์นวัตกรรมรุ่นใหม่ โดยเน้นให้ผู้เรียนได้เข้าใจภาพรวมของ Arduino ตั้งแต่จุดเริ่มต้นไปจนถึงตัวอย่างการสร้างนวัตกรรมสิ่งประดิษฐ์ใหม่ๆ ที่น่าสนใจ รู้จักบอร์ดมาตรฐานของ Arduino และอุปกรณ์เสริมประเภทต่างๆ ตลอดจนการเลือกซื้อ Arduino ให้ตรงกับงานที่ต้องการประดิษฐ์ รู้จักการใช้งานซอฟต์แวร์เพื่อการเรียนรู้ Arduino แบบจำลองทั้งการออกแบบ การต่อวงจร และการเขียนโปรแกรมเพื่อจำลองการทำงานของ Arduino แบบออนไลน์ โดยไม่จำเป็นต้องใช้บอร์ดจริงเพื่อทดสอบแนวความคิด วิธีการ และผลการทดลองก่อนลงทุนซื้ออุปกรณ์จริง

ผู้เรียนจะได้เรียนรู้การใช้งานบอร์ด Arduino จริง ตั้งแต่เริ่มใช้งานครั้งแรกไปจนถึงการแสดงผลแบบต่างๆ และส่วนที่สำคัญที่สุด คือ การเขียนโปรแกรมควบคุมการทำงานซึ่งผมได้พยายามนำเอาประสบการณ์ที่มีทางด้านการสอนการเขียนโปรแกรม มาถ่ายทอดเรียบเรียงเป็นข้อความและรูปภาพ เพื่อให้เรียนได้ด้วยตัวเองแบบเห็นภาพที่เข้าใจง่ายที่สุดโดยเจาะลึกการเขียนโปรแกรม ตั้งแต่การรู้จักกลุ่มของคำสั่งต่างๆ ของ Arduino ที่มีให้ใช้งาน วิธีการเขียนโปรแกรม Arduino Sketch การใช้งานตัวแปร ชุดอักขระ ชนิดข้อมูลและตัวแปร ตัวดำเนินการ การใช้คำสั่งทำซ้ำ การใช้คำสั่งเงื่อนไข การใช้งานอาร์เรย์ และการสร้าง Functions ด้วยตัวเอง โดยถ่ายทอดออกมาเป็นบทเรียน Arduino Tutorials ภาคปฏิบัติ ที่สามารถเขียนโปรแกรมไปพร้อมกันแบบ Step-by-Step พร้อมด้วยรูปภาพ Flowchart เพื่อให้ผู้เรียนเห็นภาพการทำงานของโปรแกรมได้ด้วยตัวเอง

ผมหวังว่า หนังสือเล่มนี้จะเป็นจุดเริ่มต้นและแรงบันดาลใจที่ดีให้กับผู้เรียน ที่อยากจะเริ่มต้น และต้องการพัฒนาทักษะการเขียนโปรแกรมของตนเองให้ดียิ่งขึ้น และสามารถนำความรู้และทักษะที่ได้ไปต่อยอดในการสร้างโครงงานนวัตกรรมสิ่งประดิษฐ์ เพื่อส่งเสริมทางด้านการเรียน การประกอบธุรกิจ และการสร้างสรรค์สังคมให้ดียิ่งๆ ขึ้นได้

อาจารย์ทศพล บ้านคลองสี่

อาจารย์มหาวิทยาลัยและวิทยากรพิเศษทั้งภาครัฐและเอกชน

เจ้าของเพจ/ยูทูบ : Aj. NesT the Series และ GlurGeek.com



EDITOR'S NOTE

การเป็นบรรณาธิการหนังสือเล่มนี้ ทำให้ผมได้มีโอกาสรื้อฟื้นความรู้ในสาขาที่จบมา นั่นคือ สาขาอิเล็กทรอนิกส์ เอ็นจิเนียริ่ง พอได้อ่านต้นฉบับจบก็ยิ่งทำให้ผมสนุกเป็นอย่างยิ่ง เพราะได้เห็นว่าอุปสรรคต่างๆ ที่ผมเคยประสบในสมัยเรียนได้ถูกขจัดออกไปเสียจนหมดสิ้น จากศาสตร์ที่เคยเป็นเรื่องเฉพาะคนที่เรียนจบมาตรงสายกลับกลายเป็นเรื่องที่เปิดกว้างสำหรับทุกๆ คน มันเป็นเช่นนั้นได้อย่างไร?

มันคงจะเป็นไปไม่ได้ถ้าไม่มีฮีโร่เหล่านี้คือ นาย Massimo Banzi และเพื่อนๆ ผู้ออกแบบบอร์ดไมโครคอนโทรลเลอร์นามว่า “Arduino” ที่ตัดเอาความซับซ้อนออกไปเพื่อให้เป็นบอร์ดที่เรียนรู้ง่าย ขจัดสิ่งที่เป็นอุปสรรคในการพัฒนาออกไปเสียจนหมดสิ้นด้วยการทำให้ฮาร์ดแวร์และซอฟต์แวร์เป็น Open Source และสร้าง Arduino Community เพื่อเป็นแหล่งสร้างแรงบันดาลใจ แบ่งปันไอเดีย ด้วยการสร้าง Arduino Project Hub ศูนย์รวมโปรเจ็กต์ที่ช่วยบ่มเพาะไอเดียชั้นดี สำหรับคนที่ยังคิดไม่ออกว่าจะประดิษฐ์อะไรดี และ Arduino Forum แหล่งแลกเปลี่ยนความรู้ระหว่างสมาชิก

ด้วยความที่เป็นกลุ่มนักออกแบบจากประเทศอิตาลี ที่ขึ้นชื่อในเรื่องของความคิดสร้างสรรค์ จึงมีแนวทางในการพัฒนา Arduino Products ที่แตกต่างจากยุคสมัยเดิม โดยได้พัฒนา Products ที่ครอบคลุมคนทุกระดับตลอดเส้นทางการศึกษา ทั้งในระดับนักเรียน นักศึกษา นักวิจัย เมกเกอร์ (Maker) ทั้งมือสมัครเล่นและมืออาชีพ มีตั้งแต่ชุดประกอบสำเร็จรูป (Kits) มีบอร์ดหลายรุ่นที่มีฟีเจอร์และฟอร์มแฟ็กเตอร์ขนาดต่างๆ เพื่อรองรับโครงงานที่มีสเกล และความซับซ้อนที่แตกต่างกัน และกลุ่มสินค้าขั้นสูง เช่น STEAM, Robotics & Drones, Internet Of Things (IoT), Industrial Automation เป็นต้น นอกจากนี้ยังมีวงจรขยายซัพพอร์ตมากมาย

หนังสือเล่มนี้เขียนโดย อาจารย์ทศพล บ้านคลองสี่ หรือ Aj. NesT the Series จะช่วยสร้างทักษะพื้นฐานทางอิเล็กทรอนิกส์ การต่อวงจร และการเขียนโปรแกรมควบคุมการทำงาน สามารถเริ่มต้นก้าวแรกได้อย่างรวดเร็วจากหนังสือ และ VDO Clip ใน YouTube Channel รวมถึงได้รับประสบการณ์จริงจากการทำ LAB ที่มีทั้งการต่อวงจรจริง และต่อวงจรจำลองผ่าน Simulation Software ผมเชื่อว่า ผู้เรียนจะได้รับประสบการณ์การเรียนรู้ที่ครบถ้วนบริบูรณ์ที่สุดจากหนังสือเล่มนี้

What is a Maker? : http://bitly.ws/oH6R

Makerspace : https://www.thekommon.co/makerspace-for-education/

ภีรพล คชาเจริญ

บรรณาธิการ

CONTENTS

Chapter 01
ภาพรวม Arduino สำหรับผู้เริ่มต้น

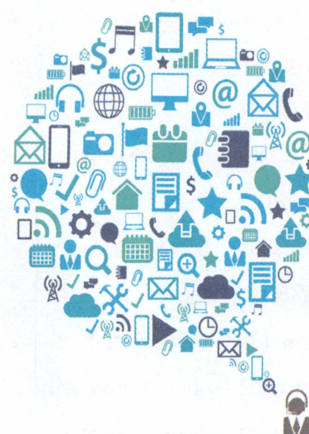
Chapter 02
รู้จักบอร์ดมาตรฐานของ Arduino และอุปกรณ์เสริม

Chapter 03
การเลือกซื้อ Arduino ให้เหมาะกับการใช้งาน

Chapter 04
ซอฟต์แวร์เพื่อการเรียนรู้ Arduino

Chapter 05
เริ่มใช้งานบอร์ด Arduino ครั้งแรก

Chapter 06 การเขียนโปรแกรมควบคุมการทำงาน

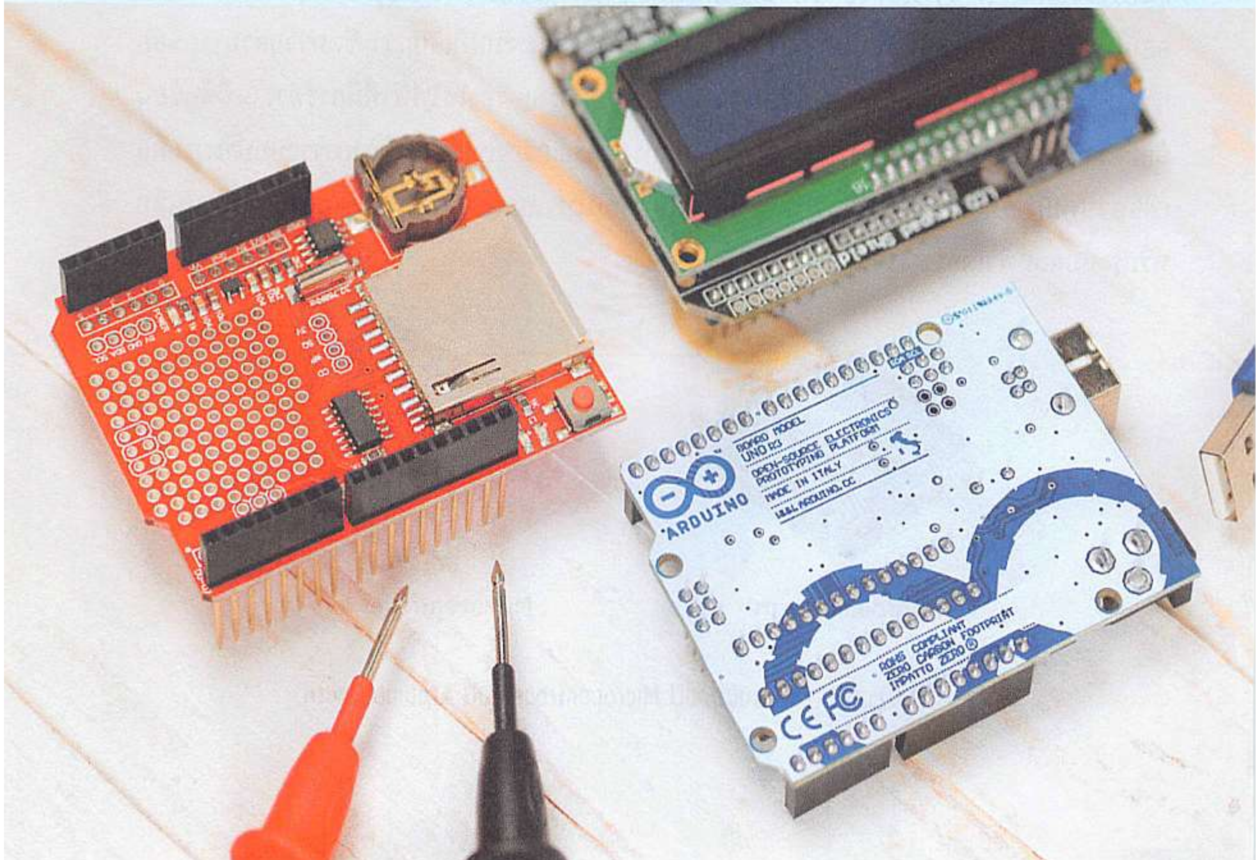


CHAPTER

01

ภาพรวม Arduino สำหรับผู้เริ่มต้น

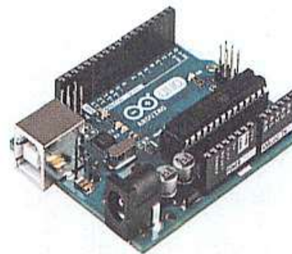
ในบทแรกนี้ ผมจะพาทุกท่านไปรู้จักกับโลกของ Arduino ในภาพกว้าง ให้รู้ว่ามันคืออะไร นำไปใช้ประโยชน์อย่างไร พาย้อนกลับไปหาต้นกำเนิดความเป็นมาของบอร์ด การเลือกใช้บอร์ด Arduino เพื่อการศึกษา การพัฒนาทักษะทางด้านอิเล็กทรอนิกส์ และการเขียนโปรแกรม องค์ประกอบของการพัฒนา Arduino Projects ตลอดจนแหล่งเรียนรู้เพิ่มเติมในรูปของคลิป VDO แสดงตัวอย่างโครงงานมากมายจากทั่วโลก



Arduino คืออะไร?

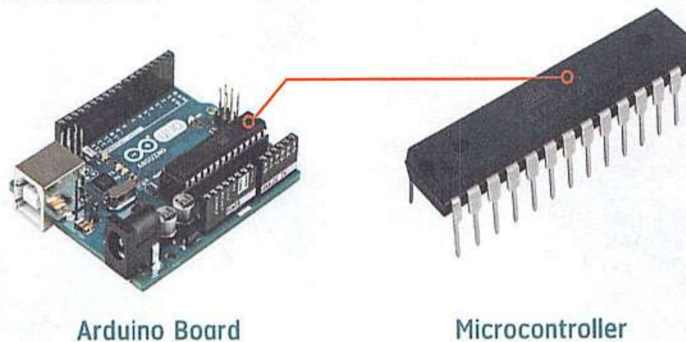
Arduino คือ บอร์ดไมโครคอนโทรลเลอร์ที่ใช้สร้างโปรเจ็กต์ที่เป็นสิ่งประดิษฐ์ที่เกี่ยวข้องในเชิงอิเล็กทรอนิกส์ (Electronics), วงจรไฟฟ้า (Circuit) และการเขียนโปรแกรม (Programming) โดยมีไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์ทั้งหมด





▲ ภาพแสดง Arduino Logo และบอร์ดรุ่นมาตรฐาน Arduino UNO

ไมโครคอนโทรลเลอร์ (Microcontroller Unit : MCU) เป็นวงจรรวมขนาดเล็กที่ถูกออกแบบมาเพื่อควบคุมการทำงานเฉพาะในระบบฝังตัว หรือระบบสมองกลฝังตัว (Embedded System) โดยทั่วไปไมโครคอนโทรลเลอร์จะประกอบด้วยโพรเซสเซอร์ (Processor), หน่วยความจำ (Memory) และส่วนเชื่อมต่ออุปกรณ์ต่อพ่วง (I/O) ติดตั้งอยู่บนชิปตัวเดียวกัน (ชิปวงจรรวม หรือไอซี) จะเห็นว่าระบบไมโครคอนโทรลเลอร์นั้นคล้ายกับระบบคอมพิวเตอร์ ต่างกันพียงเป็นระบบที่เล็กกว่า ขีดความสามารถน้อยกว่า เพราะมีหน้าที่ในการควบคุมการทำงานอยู่ในอุปกรณ์ หรือเครื่องใช้ไฟฟ้าที่มีการทำงานที่ซับซ้อนน้อยกว่านั่นเอง เราสามารถพบระบบฝังตัวได้ในอุปกรณ์ หรือเครื่องใช้ไฟฟ้าที่ต้องการระบบประมวลผลเพื่อควบคุมการทำงาน จึงอาจกล่าวได้ว่า **ไมโครคอนโทรลเลอร์ คือ ระบบคอนโทรลขนาดเล็กหรือระบบคอมพิวเตอร์ขนาดเล็ก**



▲ รูปแสดงตำแหน่งของชิป Microcontroller บน Arduino Board

คำว่า **"Arduino"** มีความหมายครอบคลุมทั้งฮาร์ดแวร์และซอฟต์แวร์ เป็นแพลตฟอร์มแบบโอเพ่นซอร์สที่มีการเผยแพร่อุปกรณ์และวงจรต้นแบบของ Arduino Board และซอร์สโค้ดของ Arduino Software (IDE) สู่สาธารณะ สามารถดาวน์โหลดได้ฟรี เพื่อให้ใครก็ได้ไม่ว่าจะเป็นนักอิเล็กทรอนิกส์ นักพัฒนาซอฟต์แวร์ โปรแกรมเมอร์ สามารถนำไปพัฒนาและประยุกต์ใช้ให้เหมาะกับความต้องการของตนเอง หรือดัดแปลงและผลิตให้ตรงกับความต้องการของกลุ่มเป้าหมายได้อย่างอิสระ จึงทำให้เกิดเป็นชุมชน Arduino Community ที่สามารถแลกเปลี่ยนข้อมูลในการพัฒนาสิ่งประดิษฐ์ได้อย่างกว้างขวาง เพราะเป็นชุมชนที่มีสมาชิกจากทั่วโลก

คลิปแนะนำ Arduino ใน 15 นาที

คลิปแนะนำ Arduino สำหรับผู้เริ่มต้น เรียนรู้วิธีการเลือกบอร์ด Arduino และตัวอย่างโครงงาน เช่น ไฟ LED หรี่แสง การสร้างตัวควบคุมความเร็วมอเตอร์ และอื่นๆ



▲ คำค้นหา "You can learn Arduino in 15 minutes"

www.youtube.com/watch?v=nL34zDTPkcs



Arduino อ่านว่าอย่างไร?

Arduino เป็นภาษาอิตาลี ถ้าถอดคำอ่านตรงตัว อ่านว่า "อา ดุ อิ โน่" หรือที่บ้านเรานิยมอ่านว่า "อา ดุย โน่" ในภาษาเขียนของหนังสือเล่มนี้จะขอใช้คำว่า "Arduino" เป็นภาษาอังกฤษ เพื่อหลีกเลี่ยงความสับสับในการสะกดคำอ่านด้วยภาษาไทย

เราสามารถฟังเสียงอ่านต้นฉบับคำว่า "Arduino" ทั้งภาษาอิตาลีและภาษาอังกฤษได้จาก Google Translate โดยเลือก Italian และ English จากนั้นคลิกที่รูปลำโพง (Speaker) ดังรูป



ทำไมบอร์ด Arduino จึงเป็นทางเลือกที่ดีที่สุด

- Arduino Software (IDE) มีส่วนติดต่อกับผู้ใช้ที่เรียบง่าย การใช้งานก็ง่าย จึงเหมาะสำหรับผู้เริ่มต้น และตัวโปรแกรมยังมีความยืดหยุ่นเพียงพอสำหรับผู้ใช้งานในระดับสูงอีกด้วย
- Arduino Software (IDE) รองรับการทำงานข้ามแพลตฟอร์ม สามารถทำงานบนระบบปฏิบัติการได้หลากหลาย ไม่ว่าจะเป็นระบบ Windows, Macintosh OS X และ Linux ในขณะที่ระบบไมโครคอนโทรลเลอร์ส่วนใหญ่จำกัดอยู่ที่ Windows เท่านั้น
- Arduino Software (IDE) เป็นซอฟต์แวร์โอเพ่นซอร์ส ซึ่งเปิดโอกาสให้โปรแกรมเมอร์ หรือนักพัฒนาซอฟต์แวร์ทั่วโลก สามารถพัฒนา Arduino Software ผ่านไลบรารีที่เขียนขึ้นด้วยภาษา C++ สำหรับใครที่รู้เรื่องทางเทคนิคมากหน่อยก็สามารถพัฒนาบน AVR ซึ่งเป็นไมโครคอนโทรลเลอร์อีกตระกูลหนึ่งแทนได้ โดยตัวนี้สนับสนุนภาษา C พอเขียนเสร็จก็เพิ่ม C Code จาก AVR ไปยัง Arduino IDE ได้เลย

- ครูและนักเรียนสามารถนำบอร์ด Arduino ไปพัฒนาเพื่อสร้างเครื่องมือทางวิทยาศาสตร์ที่มีต้นทุนต่ำ เพื่อพิสูจน์หลักการทางเคมีและฟิสิกส์ หรือเพื่อเริ่มต้นการเขียนโปรแกรมและหุ่นยนต์ พวกนักออกแบบและสถาปนิกใช้มันเพื่อสร้างต้นแบบที่สามารถโต้ตอบได้ นักดนตรีและศิลปินใช้มันเพื่อทดสอบเครื่องดนตรีชิ้นใหม่ๆ
- บอร์ด Arduino ได้รับการออกแบบโดยพยายามลดความซับซ้อนของโพรเซสการทำงานกับไมโครคอนโทรลเลอร์ลง ทำให้บอร์ด Arduino ไม่ยุ่งยากซับซ้อนเหมือนบอร์ดตัวอื่นๆ
- บอร์ด Arduino ได้รับการเผยแพร่ภายใต้ Creative Commons License หรือสัญญาอนุญาตแบบเปิด ดังนั้น ใครที่มีความชำนาญในการออกแบบวงจรสามารถสร้างบอร์ดขึ้นมาเองได้โดยขยายเพิ่มวงจรให้มีคุณสมบัติตามต้องการ หรือแม้แต่ผู้เริ่มต้นก็สามารถสร้างชิ้นงานผ่านบอร์ดทดลองที่เรียกว่า **"Breadboard"** เพื่อทำความเข้าใจวิธีการทำงานในแบบประหยัด (สร้าง Arduino บน Breadboard : www.arduino.cc/en/Main/Standalone)

กล่าวโดยสรุป Arduino นั้นเป็นเครื่องมือสำคัญในการเรียนรู้สิ่งใหม่สำหรับทุกคน ไม่ว่าจะเป็นนักเรียน นักสะสม ศิลปิน หรือโปรแกรมเมอร์ ก็สามารถเริ่มประดิษฐ์ได้โดยทำตามคำแนะนำไปทีละขั้นตอนจากชุดอุปกรณ์เรียนรู้ Arduino Starter Kit รวมทั้งการแบ่งปันไอเดียกับเพื่อนสมาชิกคนอื่นๆ ในชุมชนออนไลน์ของ Arduino ด้วยเหตุนี้ Arduino จึงเป็นบอร์ดที่ตอบโจทย์ทั้งครู นักเรียน เมกเกอร์ หรือแม้แต่บุคคลทั่วไปที่สนใจอยากฝึกทักษะทางด้านอิเล็กทรอนิกส์ การต่อวงจร และการเขียนโปรแกรมควบคุมโดยบอร์ด Arduino จะรองรับพื้นฐานเหล่านี้ทั้งหมด ผู้เริ่มต้นสามารถนำไปประยุกต์ใช้งานได้ง่ายเพื่อสร้างชิ้นงานหรือสิ่งประดิษฐ์ที่ใช้ประโยชน์ได้ในชีวิตจริง หรือจะสร้างของเล่นแปลกใหม่ที่ใช้ไมโครคอนโทรลเลอร์ในการควบคุม



Arduino Software (IDE) คืออะไร?

หรือจะเรียกว่า **"Arduino IDE"** โดย IDE ย่อมาจาก "Integrated Development Environment" คือ โปรแกรมที่มีเครื่องมืออำนวยความสะดวกในการเขียนโค้ด (Code) หรือชุดคำสั่ง เช่น คำสั่งพื้นฐานที่จำเป็น, เครื่องมือตรวจสอบ Syntax, คอนโซลแยกส่วนของโค้ดและผลลัพธ์ เป็นต้น

Arduino บอร์ดเพื่อการศึกษา





▲ Source : Intel IQ

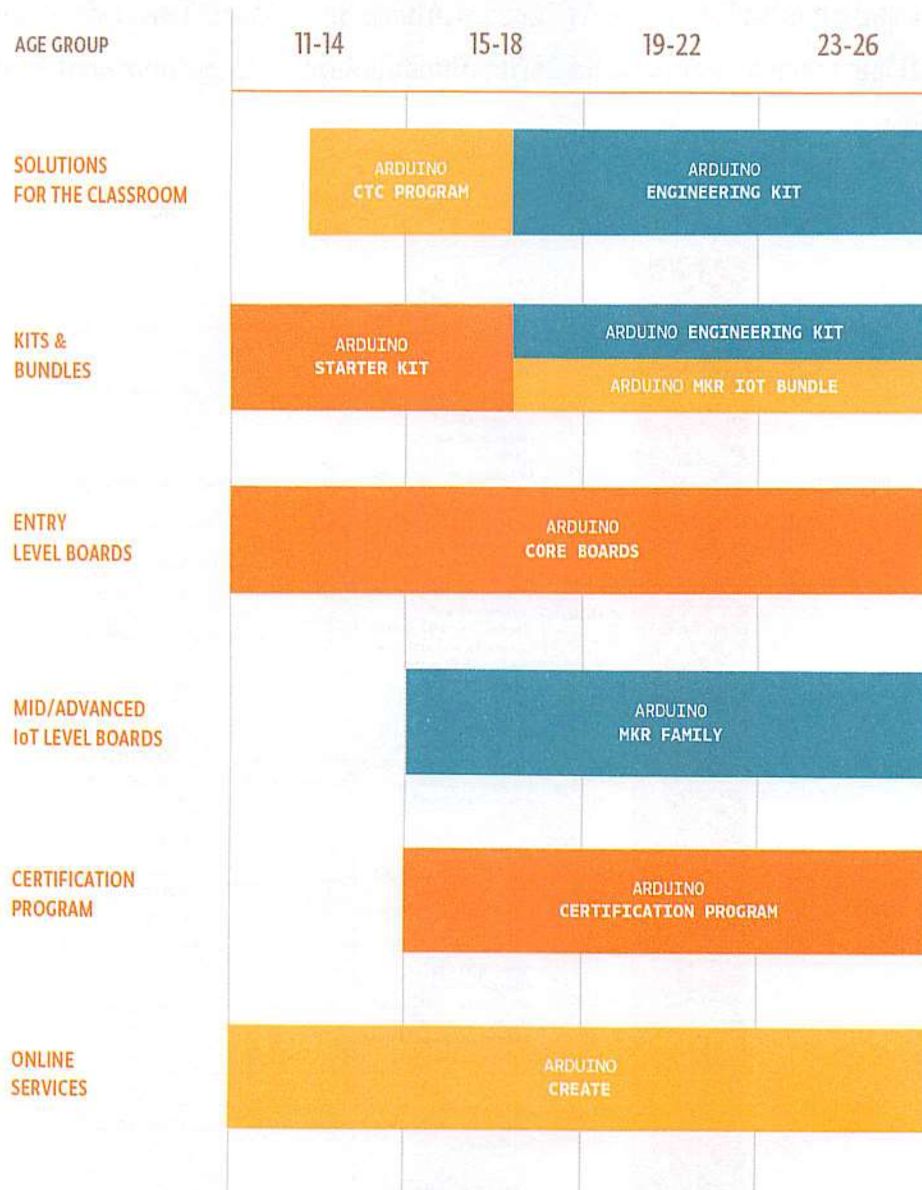
เมื่อเข้าไปศึกษาที่เว็บไซต์หลักของ Arduino จะพบว่าที่นี่มีข้อมูลที่สนับสนุนในด้านการศึกษาเรียนรู้มากมาย อย่างเช่นหัวข้อ **"Teaching, Inspiring and Empowering!"** (https://www.arduino.cc/education) แสดงให้เห็นว่า ทาง Arduino ให้ความสำคัญกับการพัฒนาผลิตภัณฑ์เพื่อสนับสนุนภาคการศึกษาเป็นอย่างยิ่ง โดยมีการก่อตั้งหน่วยงานที่ชื่อ **"Arduino Education"** ขึ้นมา อันประกอบด้วยทีมงานที่จัดตั้งขึ้นโดยผู้เชี่ยวชาญทางด้านการศึกษา นักพัฒนาเนื้อหา วิศวกร และนักออกแบบการสื่อสารจากทั่วทุกมุมโลก โดยมีเป้าหมายมุ่งเน้นการพัฒนาระบบ STEAM รุ่นใหม่ๆ เพื่อตอบสนองความต้องการของนักเรียนและนักการศึกษา ตลอดหลักสูตรการศึกษาในระดับต่างๆ

ภารกิจที่สำคัญ คือ การทำให้ทุกๆ คนสามารถเข้าถึงเทคโนโลยีได้ โดยพามันไปอยู่ในมือของนักเรียนและนักการศึกษาทุกๆ คน และเพื่อให้ภารกิจนี้เป็นจริง จึงได้ก่อตั้ง Arduino Education ขึ้นมาเพื่อปฏิบัติภารกิจนี้ให้บรรลุเป้าหมาย โดยเน้นการสร้างโปรแกรม STEAM ใหม่ๆ ที่รวมเอาวิทยาศาสตร์ เทคโนโลยี วิศวกรรม คณิตศาสตร์ และศิลปะ เพื่อรองรับความต้องการของผู้สอน และนักเรียนในภาคการศึกษาทุกระดับ

สถาบันการศึกษาหลายแห่งเลือกเข้าเว็บไซต์ Arduino.cc เพื่อเป็นแหล่งศึกษาข้อมูล ให้ความรู้แก่นักเรียนในหัวข้อต่างๆ ในหลากหลายวิธี

Develop	Arduino ได้รับการยอมรับอย่างกว้างขวางในแวดวงการศึกษาทั้งในระดับมัธยมศึกษาตอนปลาย ไปจนถึงมหาวิทยาลัย/วิทยาลัย/สถาบันวิจัย ในสาขา Engineer, Internet of Things, Robotics, Art และ Design
Teach and Learn	โรงเรียนมัธยมหลายแห่งบรรจุ Arduino ในหลักสูตรทางนวัตกรรมสำหรับการศึกษาข้ามหลักสูตร
Play	โรงเรียนประถมศึกษาใช้ของเล่นที่มีเทคโนโลยีของ Arduino ติดตั้งอยู่ภายในเพื่อแนะนำการเรียนรู้ทางกายภาพ ตรรกะ ทักษะการสร้าง และการแก้ปัญหา

Arduino Education นำเสนอทางเลือกสำหรับห้องเรียนด้วยชุดอุปกรณ์ไว้หลากหลาย ได้แก่ ชุด Starter Kit ชุดประกอบชิ้นงานสำเร็จรูป บอร์ดสำหรับศึกษาด้วยตนเอง และชุดประกอบที่ต้องทำงานเป็นกลุ่ม เป็นต้น โดยผู้เรียนจะต้องสำรวจอุปกรณ์ต่างๆ ในชุดประกอบ เพื่อดูว่ามีชิ้นส่วนอะไรบ้าง อะไรที่เกี่ยวข้องกัน หาทางที่จะประกอบชิ้นส่วนต่างๆ เข้าด้วยกัน รวมถึงการคิดที่จะประดิษฐ์เป็นโครงงานที่สามารถนำมาใช้ประโยชน์ได้จริงจากบอร์ดรุ่นต่างๆ



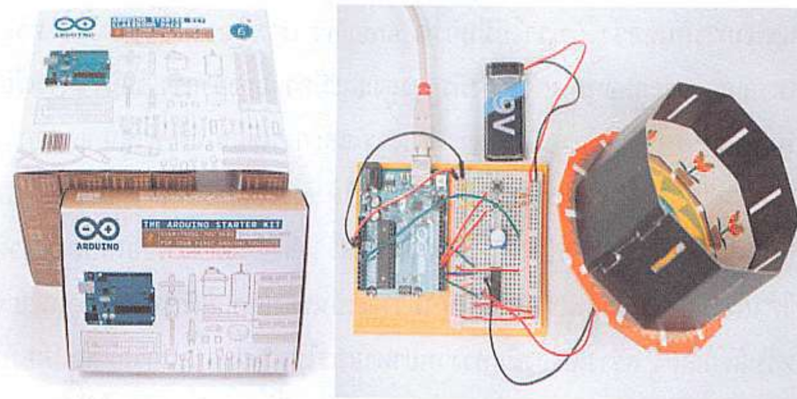
▲ รูปแสดง Arduino Products ในทุกช่วงวัยตลอดเส้นทางการศึกษา

สำหรับคุณครูหรือวิทยากรที่ยังใหม่กับอุปกรณ์อิเล็กทรอนิกส์ ที่ต้องการคำแนะนำเรื่อง**การคำนวณเชิงกายภาพ (Physical Computing) และการคิดเชิงคำนวณ (Computational Thinking)** เพื่อวางแผนการสอน ตลอดจนอาจารย์มหาวิทยาลัยที่มีความเชี่ยวชาญทางด้านอิเล็กทรอนิกส์ หรือนักวิจัยระดับสูงกว่าปริญญาตรี ทาง Arduino ก็มีผลิตภัณฑ์ Arduino Kits หรือบอร์ดระดับสูงให้เลือกนำไปใช้อีกด้วย และตารางต่อไปนี้จะแสดงรายละเอียดให้เห็นว่า เราจะเลือกผลิตภัณฑ์ Arduino Kits อย่างไรให้เหมาะกับชั้นเรียน ชุดประกอบไหนใช้บอร์ดรุ่นอะไร วัตถุประสงค์ในการเรียนรู้ ทักษะอะไรที่จะได้รับการฝึกฝน และสรุปไว้อย่างชัดเจนว่า ชุดไหนเหมาะจะใช้เรียนเป็นกลุ่มในห้องเรียน และชุดไหนสามารถศึกษาได้ด้วยตนเอง

SCHOOL LEVEL	SOLUTIONS	BOARD(S) INCLUDED	SUBJECT COVERED	SOFT SKILLS	FOR
MIDDLE SCHOOL	STARTER KIT CLASSROOM PACK	ARDUINO UNO (X6)	Fundamentals of programming and electronics, introduction to sensors and actuators, understanding digital and analog signals	Critical thinking, collaborative learning, problem solving	CLASSROOM USE
HIGH SCHOOL	CTC 101	ARDUINO 101 (X6)	Fundamentals of programming and electronics, introduction to sensors and actuators, understanding digital and analog signals, robotics, connectivity and networking	Critical thinking, collaborative learning, problem solving	CLASSROOM USE
UNIVERSITY	STARTER KIT CLASSROOM PACK	ARDUINO UNO (X6)	Fundamentals of programming and electronics, introduction to sensors and actuators, understanding digital and analog signals	Critical thinking, collaborative learning, problem solving	CLASSROOM USE
	ENGINEERING KIT	ARDUINO MKR1000	Fundamental engineering concepts, key aspects of mechatronics, MATLAB and Simulink programming, physical engineering skills	Critical thinking, problem solving	INDIVIDUAL OR CLASSROOM USE
	MKR IOT BUNDLE	ARDUINO MKR1000	Getting Started with IoT, big data, networking, connected app development	Critical thinking, problem solving	INDIVIDUAL USE
VOCATIONAL TRAINING	STARTER KIT	ARDUINO UNO	Fundamentals of programming and electronics, introduction to sensors and actuators, understanding digital and analog signals	Critical thinking, problem solving	INDIVIDUAL USE
	MKR IOT BUNDLE	ARDUINO MKR1000	Getting Started with IoT, networking, connected app development	Critical thinking, problem solving	INDIVIDUAL USE
	ENGINEERING KIT	ARDUINO MKR1000	Fundamental engineering concepts, key aspects of mechatronics, MATLAB and Simulink programming, mechanical engineering skills	Critical thinking, problem solving	INDIVIDUAL OR CLASSROOM USE



▲ รูปแสดงเมนู Arduino Kits สำหรับเลือกชุดอุปกรณ์ให้เหมาะกับความต้องการ



▲ รูปแสดง Starter Kit Classroom Pack แพ็คเกจที่เหมาะสำหรับชั้นเรียน ที่สอนการเขียนโปรแกรมและอุปกรณ์อิเล็กทรอนิกส์



▲ ชุด Arduino Science Kit Physics Lab เหมาะสำหรับเด็กมัธยมปลาย (ช่วงอายุ 11-14)

ชมคลิปสาธิตโดยใช้คำค้นหาใน YouTube : "Arduino Science Kit Physics Lab"

www.youtube.com/watch?time_continue=33&v=4-U3JcdMoe0&feature=emb_title

Arduino จากก้าวเล็กๆ สู่ความยิ่งใหญ่

เริ่มจากก้าวแรกที่ต้องการเพียงบอร์ดไมโครคอนโทรลเลอร์ ที่ทำให้เด็กได้นำไปใช้เรียนรู้งานทางด้านอิเล็กทรอนิกส์และการเขียนโปรแกรม ซึ่งก่อนที่จะมีบอร์ด Arduino เป็นเรื่องยากมากที่จะให้เด็กประถมมาเรียนเรื่องไมโครคอนโทรลเลอร์แล้วนำไปสร้างสิ่งประดิษฐ์ขึ้นมา เพราะบอร์ดไมโครคอนโทรลเลอร์ในยุคก่อนที่จะมี Arduino มีความซับซ้อน เรียนรู้ยาก ทีมผู้ก่อตั้ง คือ นาย Massimo Banzi และเพื่อนๆ จึงช่วยกันออกแบบบอร์ดไมโครคอนโทรลเลอร์ขนาดเล็กที่ชื่อว่า Arduino ที่ใช้งานง่าย เรียนรู้ได้อย่างรวดเร็วขึ้นมา และยังเผยแพร่ข้อมูลสู่สาธารณะอย่างหมดเปลือกแบบฟรีๆ

ลองจินตนาการกันต่อว่า ถ้าเราไม่มีบอร์ด Arduino เราก็ต้องไปซื้อชิป PIC หรือไม่ก็ต้องซื้อไมโครคอนโทรลเลอร์ตระกูลอื่นๆ มาใช้ ซึ่งมีราคาสูงและมีความยุ่งยากซับซ้อนในการเรียนรู้ ที่อาจจะเหมาะกับผู้ที่มีพื้นฐานทางอิเล็กทรอนิกส์ การต่อวงจร และการเขียนโค้ดเท่านั้น แต่ยากเกินไปสำหรับคนทั่วไป หรือแม้แต่เด็กชั้นประถมหรือมัธยมที่จะเข้าถึงได้ ซึ่ง Arduino ได้ทำลายข้อจำกัดเหล่านั้นลงได้ และพยายามจุดประกายให้กับทุกๆ คนได้ลุกขึ้นมาสร้างสิ่งประดิษฐ์ที่ยอดเยี่ยมเท่าที่พวกเขาจะจินตนาการได้ เกิดเป็นโปรเจ็กต์นับร้อยนับพัน ไม่ว่าจะเป็นของเด็กเล่นที่ไม่มีวางจำหน่ายที่ไหน เครื่องใช้ไฟฟ้าสำหรับสัตว์เลี้ยง สร้างเครื่องมือทางวิทยาศาสตร์ในราคาถูก อุปกรณ์เตือนภัยที่ใครๆ ก็ร่วมรายงานสถานการณ์ได้ ไปจนกระทั่งการสื่อสารกับดาวเทียมที่โคจรอยู่รอบโลกได้จริงๆ

> “ **คุณไม่จำเป็นต้องได้รับอนุญาต**
> **จากใครเพื่อสร้างสิ่งที่ยิ่งใหญ่**
>
> **คำพูดที่สร้างแรงบันดาลใจให้กับคน**
> **ทั่วโลกของ Massimo Banzi** ”

รู้จัก Arduino ผ่านการบอกเล่าของ Massimo Banzi



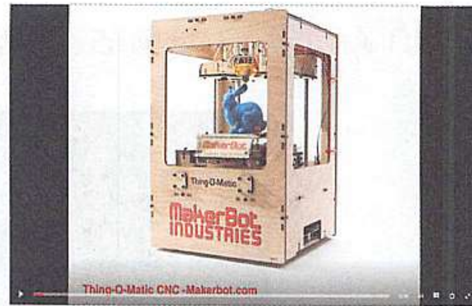
▲ ชื่อคลิป "How Arduino is open-sourcing imagination | Massimo Banzi"

This talk was presented at an official TED conference https://www.ted.com

คำบรรยายจากคลิป

ไม่กี่อาทิตย์ที่ผ่านมา เพื่อนของผมให้รถของเล่นเป็นของขวัญแก่ลูกชายวัย 8 ปี แต่แทนที่จะไปซื้อจากห้างร้านทั่วๆ ไปอย่างที่ใครๆ ทำกัน เขาเลือกที่จะเข้าเว็บไซต์ THINGIVERSE แล้วดาวน์โหลดไฟล์มาไฟล์หนึ่งแล้วพรินต์ออกมาด้วย 3D Printer เครื่องนี้ (ตามรูป) ด้วยแนวคิดที่ว่า "คุณสามารถสร้างสิ่งต่างๆ ได้ด้วยเครื่องนี้ผ่านระบบดิจิทัล" ซึ่งเป็นแนวคิดที่นิตยสาร The Economist ยกย่องให้เป็นการปฏิวัติทางอุตสาหกรรมครั้งที่ 3 ผมต้องขอแย้งว่า จริงๆ แล้วครั้งที่ 4 กำลังเกิดขึ้นด้วยซ้ำ เกิดจากฮาร์ดแวร์โอเพ่นซอร์ส และการขับเคลื่อนของนักประดิษฐ์ เพราะพรินเตอร์ที่เพื่อนผมใช้ทำของเล่นนั้นก็มีที่มาจากโอเพ่นซอร์ส โดยคุณสามารถเข้าไปในเว็บไซต์ THINGIVERSE เพื่อดาวน์โหลดทุกสิ่งทุกอย่างที่ต้องใช้ ไม่ว่าจะเป็นไฟล์ วิธีการประกอบชิ้นส่วนอุปกรณ์ รวมถึงซอฟต์แวร์ วิธีการทั้งหมดอยู่ในนั้น นี่คือส่วนหนึ่งของชุมชนที่กว้างใหญ่ ที่มีคนนับพันจากทั่วโลกกำลังสร้างพรินเตอร์แบบนี้อยู่ และมีนวัตกรรมมากมายเกิดขึ้นอยู่เรื่อยๆ เพราะมันเป็นโอเพ่นซอร์ส คุณไม่จำเป็นต้องขออนุญาตใครเพื่อสร้างสิ่งที่ยิ่งใหญ่ มันเป็นเหตุการณ์คล้ายกับคอมพิวเตอร์พีซีที่เกิดขึ้นในปี ค.ศ. 1976 ที่บริษัท Apple และบริษัทอื่นๆ กำลังแข่งขันกันพัฒนา ซึ่งในอีกไม่กี่ปีนับจากนี้เราจะได้เห็นเหตุการณ์คล้ายๆ กัน





▲ เว็บไซต์ THINGIVERSE และ 3D Printer

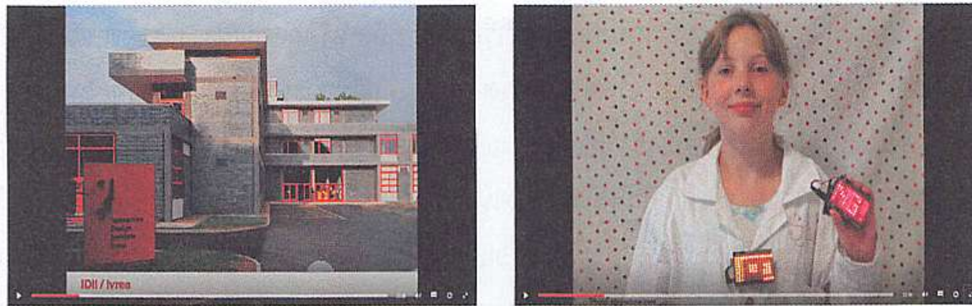
มีอีกเรื่องที่น่าสนใจครับ ผมบอกว่า วงจรอิเล็กทรอนิกส์ก็เป็นโอเพ่นซอร์ส เพราะหัวใจของพรินเตอร์เครื่องนี้คือสิ่งที่ผมผูกพันเป็นอย่างมาก นั่นคือ แผงวงจร Arduino ที่ทำหน้าที่ขับเคลื่อนเจ้าพรินเตอร์ มันเป็นโปรเจ็กต์ที่ผมทำมาตลอด 7 ปี เป็นโปรเจ็กต์โอเพ่นซอร์สที่ผมทำร่วมกับเพื่อนๆ 5 คน (ดังรูป) มีชาวอเมริกัน 2 คน อิตาเลียน 2 คน และชาวสเปนอีก 1 คน มันเป็นโปรเจ็กต์ระดับโลกน่ะครับ





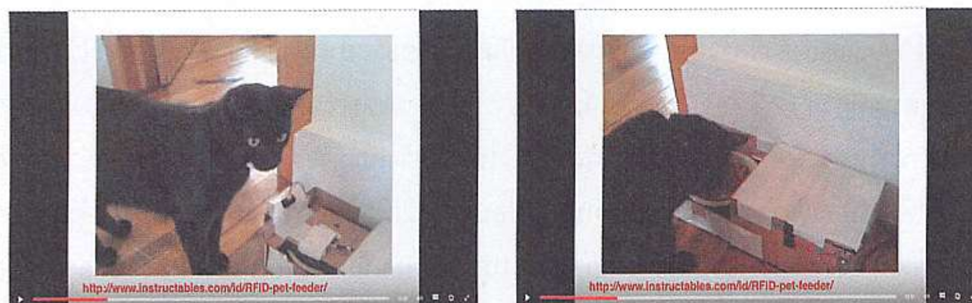
▲ กลุ่มผู้บุกเบิกในการออกแบบและสร้างบอร์ด Arduino เพื่อการศึกษาอย่างแท้จริง

เรารวมตัวกันที่สถาบันด้านการออกแบบชื่อว่า Interaction Design Institute ในเมืองอีเวีย ประเทศอิตาลี ซึ่งสอนการออกแบบที่ตอบสนองการใช้งาน ซึ่งหมายถึงการออกแบบรูปทรงพื้นฐานของวัตถุหนึ่งๆ แล้วต่อยอดออกไป เป็นงานดีไซน์ที่เข้ากับวิธีการที่เราโต้ตอบกับสิ่งต่างๆ เวลาเราออกแบบวัตถุที่ต้องโต้ตอบกับมนุษย์ ถ้าเราทำมือถือจำลองจากโฟมมันก็คงดูไม่เข้าท่าสักเท่าไร คุณต้องทำอะไรที่สามารถโต้ตอบกับมนุษย์ได้จริงๆ เราจึงหันมาทำ Arduino และโปรเจ็กต์อื่นๆ มากมาย เพื่อสร้างต้นแบบที่ง่ายสำหรับให้นักเรียนของเราใช้กัน เพื่อให้นักเรียนได้สร้างสิ่งที่ทำงานได้จริงๆ โดยไม่จำเป็นต้องใช้เวลา 5 ปี เพื่อศึกษาด้านวิศวกรรมไฟฟ้า เราใช้เวลาเพียงเดือนเดียว แล้วเราสร้างสิ่งที่แม้กระทั่งเด็กก็สามารถนำไปใช้ได้ และด้วย Arduino ที่พวกเราสร้างขึ้นนี่เอง เราพบว่าเด็กอย่างน้อง Sylvia สามารถสร้างโปรเจ็กต์จาก Arduino ได้จริงๆ ผมถึงกับตกใจเมื่อได้เห็นเด็กอายุ 11 ปี ที่เอาผลงานที่เธอสร้างจาก Arduino มาให้ผมดู ผมว่ามันน่ากลัวมากนะ มันทำให้ผมเห็นศักยภาพที่เด็กเหล่านี้มีเมื่อเราให้เครื่องมือเหล่านี้แก่พวกเขา



▲ สถาบันการออกแบบ Interaction Design Institute และน้อง Sylvia

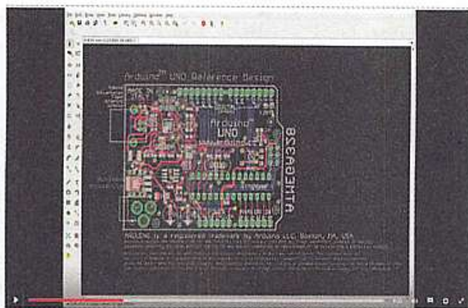
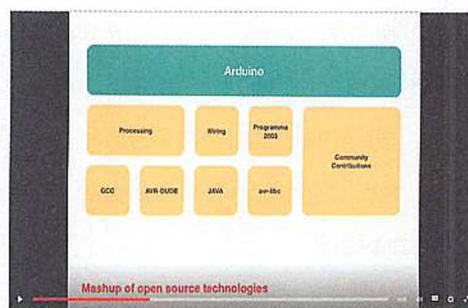
เรามาดูกันว่า เกิดอะไรขึ้นเมื่อเรามีเครื่องมือที่ใครก็ได้สามารถหยิบขึ้นมา และสร้างอะไรสักอย่าง นี่คือตัวอย่างหนึ่งที่ผมชอบใช้ในการเริ่มบทสนทนา ตัวอย่างของเครื่องให้อาหารแมว สุภาพบุรุษผู้สร้างโปรเจ็กต์นี้เลี้ยงแมวไว้ 2 ตัว ตัวหนึ่งป่วย อีกตัวหนึ่งสบายดี เขาเลยต้องมั่นใจว่า ทั้งคู่จะได้รับอาหารอย่างเพียงพอ โดยประดิษฐ์เครื่องมือแยกแยะแมวจากชิปที่ฝังอยู่ในปลอกคอของแมวแต่ละตัว และจะเปิดประตูเพื่อให้แมวตัวนั้นได้กินอาหาร เครื่องมือนี้ถูกสร้างขึ้นจากเครื่องเล่นซีดีที่ถอดออกมาจากคอมพิวเตอร์ที่ไม่ใช้แล้ว และกล่องกระดาษ เทปกาว เซนเซอร์ และหลอดไฟกะพริบ LED แล้วคุณก็ได้เครื่องมือชิ้นใหม่ออกมา ซึ่งเป็นเครื่องมือที่หาไม่ได้ตามท้องตลาด



▲ เครื่องให้อาหารแมวที่สร้างจากเครื่องเล่น CD และบอร์ด Arduino

ผมชอบวลีนี้นะ “เกาให้ถูกที่คัน” ถ้าคุณมีไอเดีย คุณต้องลองทำมันเลย เหมือนการวาดรูปลงบนกระดาษ แต่ในที่นี้เราใช้วงจรอิเล็กทรอนิกส์ หนึ่งในจุดเด่นที่ผมคิดว่าสำคัญเกี่ยวกับงานของผม คือฮาร์ดแวร์ของเรา นอกจากเราจะสร้างมันด้วยจิตวิญญาณของคนอิตาลี ดังที่คุณเห็นได้จากแผ่นวงจร (ของแท้จะสกรีนว่า “Made in Italy” ไว้ใต้บอร์ด) นั่นคือ มันเป็นอุปกรณ์โอเพ่นซอร์ส เราเปิดเผยการออกแบบแผงวงจรทั้งหมดในรูปของไฟล์ เพื่อให้คุณดาวน์โหลดและนำไปสร้างสรรค์ต่อ พัฒนาต่อยอดการทำงานเพื่อทำโปรเจ็กต์ หรือใช้ในภาคการศึกษาตามหลักสูตร

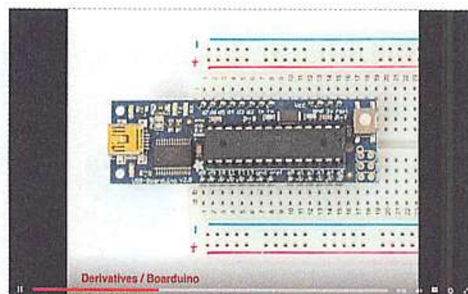
ในสมัยที่ผมเรียนโปรแกรมมิ่งใหม่ๆ ผมเองก็เรียนรู้จากโค้ด หรือดูจากแผงวงจรของคนอื่นเช่นกัน ดูตามแมกกาซีนบ้าง มันเป็นวิธีเรียนรู้ที่ดีโดยการดูงานของคนอื่น ดังนั้น ชิ้นส่วนต่างๆ ในโปรเจ็กต์ล้วนแล้วแต่เป็นแบบโอเพ่น ตัวฮาร์ดแวร์เองก็ถูกเผยแพร่ในแบบ Creative Commons License ผมชอบแนวคิดที่ว่า "ฮาร์ดแวร์กลายเป็นส่วนหนึ่งของวัฒนธรรมแห่งการแบ่งปัน" โดยร่วมมือกันพัฒนาไปเรื่อยๆ ประหนึ่งเป็นบทกลอนที่ประพันธ์ร่วมกัน หรือแม้กระทั่งซอฟต์แวร์ก็เป็นแบบ GPL หนังสือคู่มือและการเรียนการสอนก็เป็นโอเพ่นซอร์สภายใต้ Creative Commons License เพื่อที่จะรักษาชื่อ Arduino ไว้ให้เราบอกคนอื่นๆ ได้ว่า อะไรคือ Arduino และอะไรที่ไม่ใช่

▲ Arduino เป็นโอเพ่นซอร์สภายใต้ Creative Commons License

บอร์ด Arduino เองก็สร้างขึ้นจากชิ้นส่วนที่เป็นโอเพ่นซอร์สมากมาย ที่ช่วงแรกอาจจะใช้งานยากเกินไปสำหรับเด็กวัย 12 ปี Arduino ช่วยให้เราสามารถปะติดปะต่อเทคโนโลยีโอเพ่นซอร์สหลายๆ อย่างที่ทำให้สามารถประดิษฐ์อะไรสักอย่างได้ง่ายขึ้น ในบางครั้งคุณอาจตกอยู่สถานการณ์อย่างชาวชิลีคนหนึ่งที่ตัดสินใจสร้างวงจรขึ้นมาเองแทนที่จะซื้อมา ทั้งนี้เพื่อจัดเวิร์คช็อปและลดค่าใช้จ่าย มีบางบริษัทที่พัฒนา Arduino เองขึ้นมาด้วยการปรับปรุงวงจรเล็กน้อย เพื่อให้เข้ากับความต้องการของตลาดที่แตกต่างกัน ถึงตอนนี้น่าจะมีประมาณ 150 แบบเห็นจะได้





▲ บอร์ด DIY ที่เกิดจากการดัดแปลงจากบอร์ด Arduino ต้นแบบ

อันนี้สร้างโดยบริษัท Adafruit ซึ่งบริหารโดยสุภาพสตรีชื่อ Limor Fried หรือที่นิยมเรียกเธอว่า LadyAda หนึ่งในฮีโร่ของการขับเคลื่อนฮาร์ดแวร์โอเพ่นซอร์ส และการผลักดันของนักประดิษฐ์ (Maker) และนี่คือไอเดียที่ว่าเรามีสังคมแห่งการ DIY ที่มีการเติบโตอย่างรวดเร็ว มีความเชื่อมั่นในวิถีโอเพ่นซอร์ส โดยมีการร่วมมือกันทั้งบนโลกออนไลน์ และพื้นที่ต่างๆ เช่น ผลิตนิตยสารชื่อว่า **"Make:"** ที่ดึงคนเหล่านี้เข้ามาอยู่ด้วยกัน ร่วมเป็นสังคมเดียวกัน คุณจะพบโปรเจ็กต์ที่ซับซ้อนที่ถูกอธิบายด้วยภาษาง่ายๆ หรือในเว็บไซต์ instructables.com ชุมชนของคนชอบสอนหรือแนะนำให้ทำสิ่งต่างๆ มีสอนทำโปรเจ็กต์ Arduino ด้วย หน้าที่คุณเห็นอยู่บนจอนี้ จริงๆ แล้วคุณสามารถเรียนรู้วิธีทำเค้กและอื่นๆ อีกมากมาย



▲ LadyAda ผู้มีบทบาทในการขับเคลื่อนฮาร์ดแวร์โอเพ่นซอร์ส และ Make: นิตยสารสำหรับนักประดิษฐ์

เรามาดูโปรเจ็กต์บางตัวดีกว่าครับ เป็นเฮลิคอปเตอร์จำลอง 4 ใบพัด ในมุมหนึ่งมันเป็นของเล่นใช่ไหมครับ จริงๆ มันเป็นเทคโนโลยีทางทหารเมื่อไม่กี่ปีที่แล้ว แต่ตอนนี้มันเป็นโอเพ่นซอร์สที่ใช้งานง่ายและสามารถซื้อได้ทางออนไลน์ มีกลุ่มคนภายใต้ชื่อ DIY Drones กำลังสร้างสิ่งที่เรียกว่า **"Arducopter"** และมีคนที่ตั้งบริษัทขึ้นมาจริงๆ ชื่อว่า Matternet เขาสามารถใช้สิ่งนี้ในการขนส่งสิ่งของจากหมู่บ้านหนึ่งไปสู่หมู่บ้านหนึ่งในประเทศแอฟริกา และด้วยความเป็นโอเพ่นซอร์สที่เข้าถึงง่ายและดัดแปลงได้ ทำให้เขาสามารถสร้างต้นแบบบริษัทได้อย่างรวดเร็ว



▲ Arducopter ที่ถูกพัฒนาโดยบริษัท Matternet เพื่อใช้ขนส่งสิ่งของในประเทศแอฟริกา

Matt Richardson : ผมเบื่อกับการที่ต้องฟังเรื่องคนเดิมๆ บนทีวีซ้ำแล้วซ้ำอีก ผมเลยตัดสินใจทำอะไรสักอย่าง โปรเจ็กต์ Arduino นี้ ผมเรียกมันว่า "พอกันที" ซึ่งจะปิดเสียงทีวีเมื่อมีการพูดถึงเรื่องที่คุณเบื่อกับมันแล้ว ผมจะโชว์ว่าผมทำมันขึ้นมาอย่างไร

Massimo Banzi : ลองดูนะครับ

TV : "โปรดิวเซอร์ของเราไปพบกับ Kim Kardashian เพื่อหาคำตอบว่า เธอจะใส่อะไรไป...

Massimo Banzi : เป็นไงล่ะ

Matt Richardson : เจ้าสิ่งนี้จะช่วยปกป้องหูของคุณ จากการที่ต้องทนฟังรายละเอียดเกี่ยวกับงานแต่งงานของ Kim Kardashian

Massimo Banzi : สิ่งที่น่าสนใจก็คือ แมตต์เจอชิ้นส่วนต่อขยายที่ทำให้ Arduino สามารถประมวลผลสัญญาณทีวี และเขาก็เจอโค้ดที่ใช้ส่งสัญญาณอินฟราเรดสำหรับทีวี ปะติดปะต่อมันเข้าด้วยกันเพื่อสร้างโปรเจ็กต์นี้





▲ Matt Richardson ผู้ประดิษฐ์อุปกรณ์ตัดเสียงจากทีวีเมื่อมีการพูดถึงเรื่องที่คุณไม่ต้องการฟัง

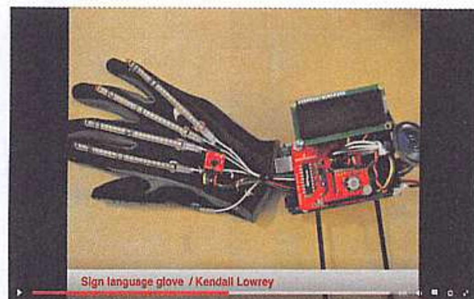
นอกจากนี้ Arduino ยังถูกใช้ในเรื่องสำคัญๆ อย่างเช่น เครื่องชนอนุภาคขนาดใหญ่ (Large Hadron Collider) มีลูกบอล Arduino ที่ใช้เก็บข้อมูล และวัดผลตัวแปรต่างๆ



หรือในกรณีของ... (เสียงดนตรี) นี่เป็นอุปกรณ์รับข้อมูลดนตรี สร้างโดยนักเรียนคนหนึ่งในอิตาลี และเขากำลังเปลี่ยนสิ่งนี้ให้กลายเป็นผลิตภัณฑ์ เป็นโปรเจ็กต์ในห้องเรียนที่กำลังจะกลายเป็นผลิตภัณฑ์



หรือการสร้างเครื่องมือช่วยเหลือคน นี่เป็นถุงมือที่เข้าใจภาษามือ และแปลท่าทางการใช้มือให้กลายเป็นเสียง และเขียนคำที่คุณกำลังสื่อด้วยภาษามือลงบนหน้าจอ แน่นอนว่าชิ้นส่วนทุกชิ้นสามารถหาได้จากเว็บไซต์ที่ขายบอร์ดและอุปกรณ์ Arduino ที่คุณสามารถประกอบขึ้นเป็นโปรเจ็กต์



หรือโปรเจ็กต์จากหลักสูตร ITP มหาวิทยาลัยนิวยอร์ก ที่เขาพบเด็กคนนี้ที่มีความพิการอย่างรุนแรง ไม่สามารถเล่นเกม PS3 ได้ พวกเขาเลยสร้างอุปกรณ์นี้ที่ทำให้เด็กสามารถเล่นเกมเบสบอลได้ ทั้งๆ ที่เด็กมีปัญหาด้านการเคลื่อนไหวร่างกาย





หรือจะเป็นโปรเจ็กต์ด้านศิลปะ อันนี้เรียกว่า **"txtBomber"** คุณป้อนข้อความลงในเครื่อง แล้วทาบอุปกรณ์ไปบนกำแพง ซึ่งจะมีขดลวดแม่เหล็กไฟฟ้าที่คอยกดปุ่มบนกระป๋องสีสเปรย์ แล้วมันก็จะฉีดพ่นสีลงบนกำแพงตามข้อความที่คุณต้องการ



เรามีต้นไม้ที่เรียกว่า **"Botanicalls"** เพราะในต้นไม้มีลูกบอล Arduino ที่เชื่อมต่อ Wi-Fi ได้ ซึ่งคอยวัดความเป็นอยู่ของต้นไม้ต้นนั้น และเชื่อมต่อกับ Twitter ทำให้คุณสามารถติดต่อสื่อสารกับมันได้ ต้นไม้ต้นนี้จะบ่นให้เราฟังได้ว่า "ร้อนจังเลย" หรือบอกว่า "หิวน้ำจังเลย" เจ้าเครื่องนี้ได้ทำให้ต้นไม้มีบุคลิกของมันเอง



หรือโปรเจ็กต์นี้ที่จะส่งทวีตเองเมื่อลูกในท้องของคุณเตะผนังหน้าท้องคุณแม่เข้า

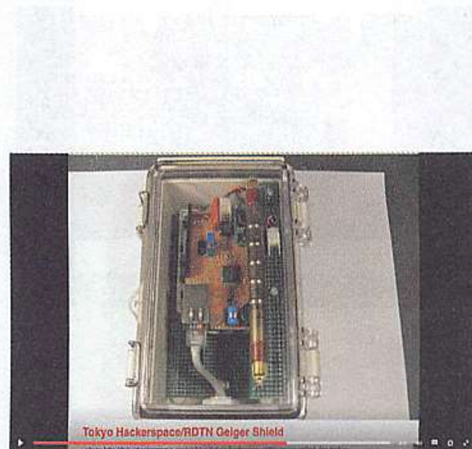


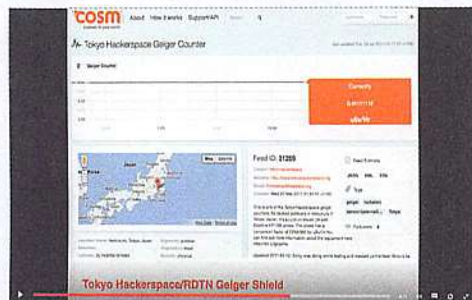
หรือโปรเจ็กต์ที่เด็กอายุเพียง 14 ปี ในประเทศชิลี ได้สร้างระบบตรวจจับแผ่นดินไหว และประกาศออกทาง Twitter เขามีคนติดตาม 280,000 คน และเด็กคนนี้กำลังจะมีโปรเจ็กต์ร่วมกับรัฐบาลในเร็วๆ นี้

หรืออีกโปรเจ็กต์ที่วิเคราะห์ข้อความทวีตของสมาชิกในครอบครัว และจะบอกคุณว่าพวกเขาอยู่ที่ไหนเหมือนที่เห็นในหนังแฮร์รี่ พอตเตอร์เลย คุณสามารถอ่านรายละเอียดทั้งหมดได้จากในเว็บไซต์ บางคนสร้างเก้าอี้ที่จะส่งทวีตทุกครั้งที่มีคนตดใส่มัน มันน่าสนใจนะครับที่ในปี 2009 เว็บไซต์ Gizmodo ได้ให้คำจำกัดความไว้ว่า โปรเจ็กต์นี้จะทำให้ทวิตเตอร์มีความหมาย อะไรๆ ก็เปลี่ยนไปเยอะนะผมว่า

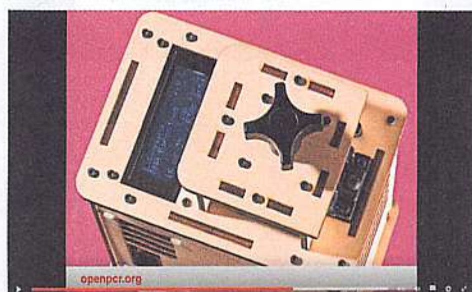


มาดูโปรเจ็กต์ที่ซีเรียสๆ กันบ้างนะครับ เมื่อครั้งภัยพิบัติที่ฟุกุชิมะ (Fukushima) มีคนญี่ปุ่นบางกลุ่มที่เล็งเห็นว่า ข้อมูลที่รัฐบาลเผยแพร่นั้นยังไม่ครบถ้วนหรือเชื่อถือได้ขนาดนั้น พวกเขาจึงสร้างเครื่องวัดไกเกอร์เคาน์เตอร์ (Geiger Counter) เชื่อมต่อกับ Arduino และอินเทอร์เน็ต พวกเขาสร้างมันขึ้นมา 100 ชุด แล้วส่งไปให้ผู้คนทั่วทั้งญี่ปุ่น และข้อมูลก็ถูกส่งขึ้นไปยังเว็บไซต์ที่ชื่อว่า **"Cosm"** ที่พวกเขาสร้างขึ้น ทำให้คุณสามารถหาข้อมูลที่เชื่อถือได้ และอัพเดตล่าสุดจากภาคสนามจริงๆ เป็นข้อมูลที่ไม่ถูกบิดเบือน





หรือเจ้าเครื่องนี้ที่มาจากการเคลื่อนไหว DIY ด้านชีววิทยา ซึ่งเป็นขั้นตอนหนึ่งที่ต้องทำเพื่อวิเคราะห์ DNA และแน่นอนว่าทั้งหมดนี้เป็นโอเพ่นซอร์ส





หรือนักเรียนในประเทศกำลังพัฒนาที่สร้างเครื่องมือมาใช้แทนเครื่องมือทางวิทยาศาสตร์ที่มักจะแพงมาก โดยพวกเขาสามารถสร้างได้ในราคาที่ถูกกว่ามากๆ โดยใช้ Arduino และอุปกรณ์เสริมอื่นๆ อันนี้เป็นเครื่องวัดความเป็นกรดด่าง



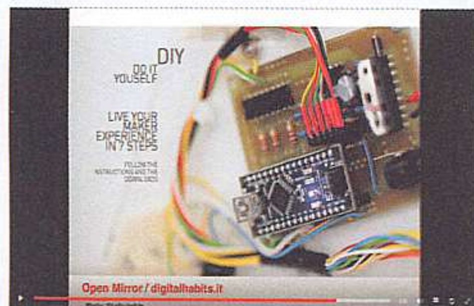
เด็กๆ ชาวสเปนเหล่านี้เรียนรู้การเขียนโปรแกรมและสร้างหุ่นยนต์ ตอนที่พวกเขาอายุประมาณ 11 ปี และใช้ Arduino ในการสร้างหุ่นยนต์ที่เตะฟุตบอลจนพวกเขากลายเป็นแชมป์โลกในด้านการสร้างหุ่นยนต์จาก Arduino และเมื่อไรที่เราต้องการสร้างหุ่นยนต์เพื่อใช้ในการศึกษาของเราเอง เราก็ไปหาพวกเขา เราบอกเด็กๆ ว่า ออกแบบให้เราหน่อยเพราะพวกเธอรู้ว่าหุ่นยนต์แบบไหนที่จะทำให้เด็กๆ สนใจ ซึ่งไม่ใช่ผมแน่ๆ เพราะผมแก่แล้ว



บริษัท Google คิดค้นเทคโนโลยีเพื่อเชื่อมต่อระหว่างมือถือ แท็บเล็ต และความเคลื่อนไหวในโลกจริงๆ มีการผลิตออกมาเป็นสินค้าชื่อ **"Accessory Development Kit"** (โอเพ่นซอร์ส) ที่มี Arduino เป็นส่วนประกอบ คลิปนี้เป็นเขาวงกตขนาดใหญ่ เขาวงกตนี้จะขยับไปตามการเคลื่อนไหวของแท็บเล็ต

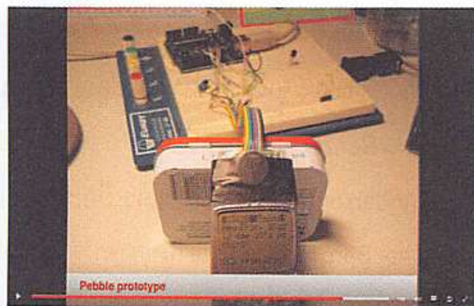
ผมมาจากอิตาลี และที่อิตาลีการออกแบบนั้นสำคัญ และยังเป็นอะไรที่อนุรักษ์นิยมมาก เราทำงานร่วมกับสตูดิโอออกแบบชื่อว่า **"Habits"** ที่มิลาน โปรเจ็กต์นี้เป็นทั้งกระจกและลำโพงสำหรับไอพอด ทั้งฮาร์ดแวร์ ซอฟต์แวร์ การออกแบบผลิตภัณฑ์ การผลิตทุกอย่างเกี่ยวกับโปรเจ็กต์นี้เป็นโอเพ่นซอร์สทั้งหมด และคุณสามารถทำได้เช่นกัน เราต้องการให้นักออกแบบคนอื่นๆ หยิบสิ่งนี้ขึ้นมาเพื่อเรียนรู้วิธีสร้าง วิธีทำผลิตภัณฑ์ที่ตอบสนองผู้คนได้ โดยเริ่มจากสิ่งที่มีอยู่จริงแล้ว


Open Mirror / digitalhabits.it



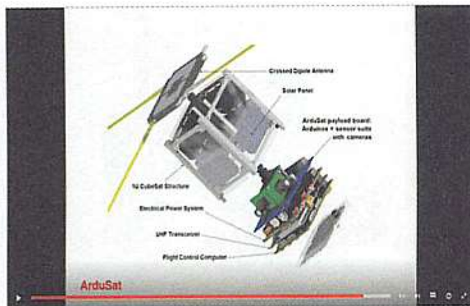
Open Mirror / digitalhabits.it

มีคนกลุ่มหนึ่งตั้งบริษัทชื่อว่า **"Pebble"** ที่สร้างต้นแบบของนาฬิกาที่เชื่อมต่อกับมือถือผ่านสัญญาณบลูทูธ สามารถแสดงข้อมูลได้ และเขาทำตัวต้นแบบจากจอ LCD เก่าๆ ของมือถือโนเกีย และ Arduino เมื่อเขาสร้างต้นแบบสำเร็จแล้ว เขาเข้าไปที่เว็บไซต์ **Kickstarter** เพื่อระดมทุนได้เงินมา 100,000 ดอลลาร์ เพื่อผลิตนาฬิกาจำหน่าย ในที่สุดพวกเขาสามารถสร้างรายได้มากถึง 10 ล้านดอลลาร์ และได้ก่อตั้งบริษัทขึ้นมาใหม่ โดยไม่ต้องขอเงินลงทุนจากใครเลย


Pebble prototype



Pebble



โปรเจ็กต์สุดท้ายที่ผมอยากแสดงให้คุณเห็น เรียกว่า **"ArduSat"** ซึ่งอยู่ใน Kickstarter ณ เวลานี้ ถ้าคุณอยากสนับสนุนก็เข้าไปในเว็บไซต์ได้เลยครับ มันเป็นดาวเทียมที่จะถูกส่งไปอวกาศจริง ๆ ซึ่งไม่น่าจะมีคนทำเป็นโอเพ่นซอร์สเลย คุณว่าไหม และมันมี Arduino ที่เชื่อมต่อกับเครื่องวัดต่าง ๆ นานา ดังนั้น ถ้าคุณใช้ Arduino เป็น คุณสามารถอัพโหลดการทดลองของคุณเข้าไปในดาวเทียมนี้ แล้วลองทดสอบดู ลองจินตนาการดูสิ หากมีเด็กมัธยมปลาย มีเวลาสักหนึ่งอาทิตย์กับเจ้าดาวเทียมนี้ เพื่อทำการทดลองนอกโลก

ดังที่ผมพูดไป มีตัวอย่างมากมาย และผมคงกล่าวได้ไม่หมด ผมอยากขอบคุณสังคม Arduino ที่ทำหน้าที่อย่างเต็มที่ สร้างสรรค์สิ่งมหัศจรรย์ออกมามากมาย ขอบคุณครับ (เสียงปรบมือ) ขอบคุณทุกคนครับ

Chris Adderson : คุณมาสซิโมครับ คุณบอกผมก่อนหน้านี้ว่า คุณไม่รู้เลยว่า มันจะยิ่งใหญ่ขนาดนี้

Massimo Banzi : ถูกต้องครับ

Chris Adderson : คุณรู้สึกอย่างไรเมื่อเห็นปรากฏการณ์นี้ และเห็นสิ่งที่คุณได้ทำให้เกิดขึ้น

Massimo Banzi : มันเป็นงานของคนจำนวนมากครับ เราเป็นชุมชนที่ทำให้ทุกคนสามารถสร้างสิ่งที่มหัศจรรย์ ผมรู้สึกตื่นตันใจมาก มันเป็นความรู้สึกที่ยากจะบรรยายนะ ทุกๆ เช้าผมตื่นขึ้นมา และดูสิ่งที่ Google Alerts ส่งมาให้ผม มันเจ๋งมากจริงๆ มันแพร่ไปยังวิทยาการทุกๆ สาขาที่เราสามารถจินตนาการได้

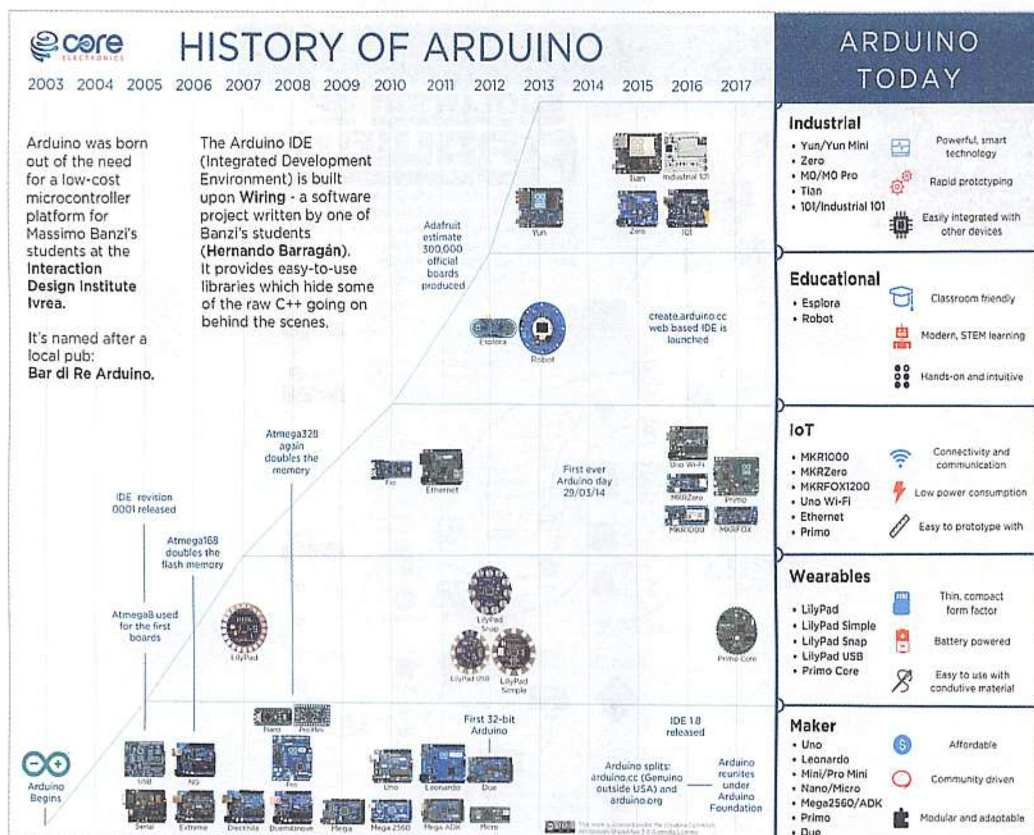
Chris Adderson : ขอบคุณมากๆ เลยครับ (เสียงปรบมือ)



◀ รูปแสดงการสนทนาในช่วงท้ายของ Massimo Banzi (ซ้าย) และ Chris Adderson (ขวา)

ย้อนประวัติของ Arduino

มีบทความชื่อ **"History of Arduino"** (July 15, 2017) ปรากฏบนหน้าเว็บของบริษัท Core Electronics (ประเทศออสเตรเลีย) สรุปการผลิตบอร์ด Arduino รุ่นต่างๆ เป็นภาพ Infographic ช่วยให้เห็นถึงประวัติของ Arduino และวิวัฒนาการของฮาร์ดแวร์ Arduino จากแรกเริ่มจนถึงปี ค.ศ. 2017 จะพบว่าบอร์ด Arduino ถูกผลิตขึ้นมาแล้วจำนวนมาก แต่ในการผลิต Arduino ขึ้นมาในแต่ละรุ่นนั้น มีการกำหนดกลุ่มเป้าหมายไว้อย่างชัดเจนว่า รุ่นไหนผลิตออกมาเพื่ออะไร โดยมีการจำแนกบอร์ดตามลักษณะการใช้งาน และยังบอกด้วยว่าบอร์ดไหนยกเลิกการผลิตไปแล้ว



▲ รูปแสดงประวัติของ Arduino ที่รวบรวมและจัดทำโดยบริษัท Core Electronics

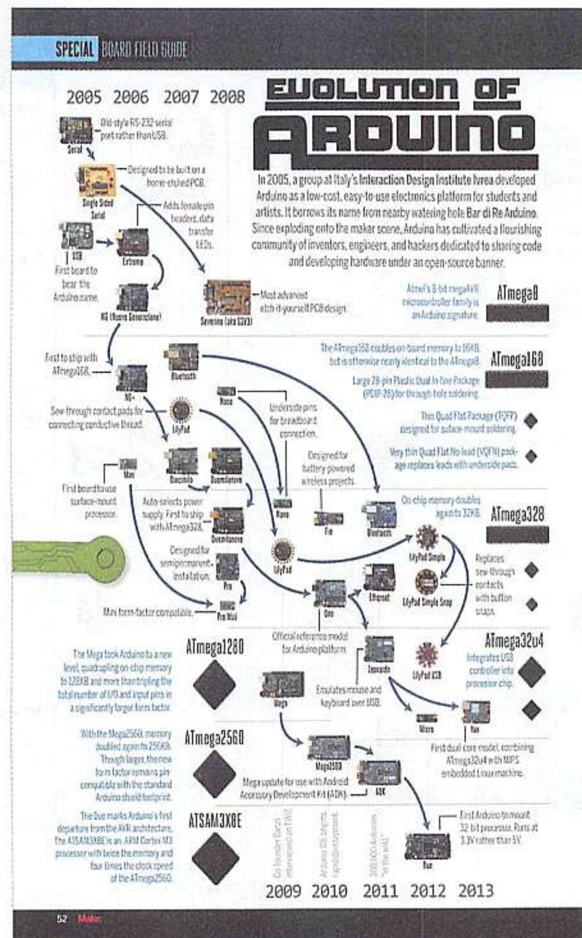
https://core-electronics.com.au



วิวัฒนาการของบอร์ด Arduino

อีกบทความหนึ่งใน **นิตยสาร MAKE** ฉบับที่ 36 ที่แม้จะออกมานานแล้วแต่ก็นำเสนอได้น่าสนใจ คือ **"Evolution of Arduino: the Family Tree"** (November 2, 2013) โดยบทความนี้ได้ย้อนประวัติกลับไปตั้งแต่บอร์ดต้นแบบของ Arduino เริ่มตั้งแต่ปี ค.ศ. 2005 และบอร์ดที่ทยอยผลิตออกมาจนถึงปี ค.ศ. 2013 (ไม่รวมบอร์ดที่ผลิตขึ้นภายหลังนิตยสารวางจำหน่าย เช่น Arduino MKR) ทางนิตยสาร MAKE เน้นไปที่การสำรวจโลกของบอร์ดรุ่นต่างๆ และนำเอาภาพมาจัดวางเพื่อแสดงให้เห็นวิวัฒนาการของบอร์ด Arduino





▲ รูปแสดง Evolution of Arduino ที่เคยตีพิมพ์ในนิตยสาร MAKE ฉบับที่ 36

https://blog.arduino.cc/2013/11/02/evolution-of-arduino-the-family-tree

เริ่มต้นศึกษา Arduino อย่างไรดี

อยากให้เริ่มต้นง่ายๆ ด้วยการมองหาแรงบันดาลใจ หรือถ้าคิดอะไรไม่ออกว่าจะประดิษฐ์คิดค้นโครงงานอะไรดี ขาดไอเดียที่จะลงมือทำ หรือยังไม่รู้ว่า Arduino สามารถนำมาสร้างชิ้นงานอะไรได้บ้าง ขอแนะนำให้เข้าไปที่ **Arduino Project Hub** ซึ่งเป็นแหล่งรวบรวมตัวอย่างโครงงานที่สร้างด้วยบอร์ด Arduino และ Sensors แบบต่างๆ หรือจะลองค้นหาใน YouTube ก็เป็นอีกทางเลือกหนึ่ง การได้ดูตัวอย่างโครงงานจากแหล่งต่างๆ จะช่วยเปิดโลกทัศน์ให้เราเห็นอะไรมากขึ้น และช่วยให้เรามีแนวทางที่จะเริ่มต้นทำโครงงาน หรือประดิษฐ์อะไรสักอย่างที่เรากำลังสนใจ

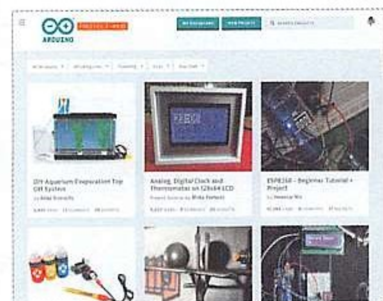
ในหัวข้อนี้จะขอพาผู้อ่านไปศึกษาโครงงานตัวอย่างจากคลิป VDO ซึ่งจำเป็นจะต้องเชื่อมต่ออินเทอร์เน็ต จะดูผ่านมือถือ แท็บเล็ต หรือพีซีก็ได้

ศึกษา Arduino Projects ใน Arduino Project Hub

Project Hub แหล่งรวบรวมผลงานของ Arduino สามารถดูข้อมูลเพื่อเป็นแนวทางในการสร้างสรรค์ชิ้นงานได้



1. เข้าเว็บไซต์ https://create.arduino.cc
2. คลิกเลือก **Arduino Project Hub** เพื่อเข้าสู่แหล่งรวบรวมโปรเจ็กต์
3. แนะนำวิธีเลือกหมวดหมู่ของ Project Hub
   - **All products** เลือกรุ่นของบอร์ด
   - **All categories** เลือก Project ในแนวที่สนใจ เช่น Sensors, Lights & LEDs, Audio & Sound
   - **Trending** เลือกตามกระแสความนิยม
   - **Any difficulty** เลือกตามระดับความยากง่ายของ Project (เหมาะกับ Beginner)
   - **Any type** เลือกแบบอื่นๆ ขอแนะนำหัวข้อ Getting started guide



ศึกษา Arduino Projects ใน YouTube

เข้าไปที่เว็บไซต์ www.youtube.com แล้วพิมพ์คำว่า "**arduino project**" ในช่องค้นหา หรือใช้คีย์เวิร์ดเฉพาะโดยเพิ่มคำเพื่อค้นหาโปรเจ็กต์ตามความสนใจ เช่น arduino project robot, arduino spider robot, arduino robot arm, arduino stepper motor, arduino iot project เป็นต้น

Top 10 Arduino Projects 2020

คลิป **Top 10 Arduino Projects 2020** | Mind Blowing Arduino School Projects เมื่อเข้าไปที่คลิปหลักจะพบลิงค์เชื่อมไปยังโปรเจ็กต์ต่างๆ ที่ผู้อ่านจะพบรายละเอียดและข้อมูลทางด้านเทคนิคอย่างครบถ้วน เช่น Required Hardware/Parts Required, Schematic, Circuit File, Arduino Code



▲ คำค้นหาใน YouTube :
"Top 10 Arduino Projects 2020"

01 Bluetooth Nurf Turret
http://bitly.ws/or8P

02 Gesture Controlled Robot
http://bitly.ws/or8T

03 Buzz Wire with Score Counter
http://bitly.ws/or8V

04 Otto DIY Robot
http://bitly.ws/or94

05 Mind Controlled Drone
http://bitly.ws/or97

06 Sea Shells Light Music Box
http://bitly.ws/or9a

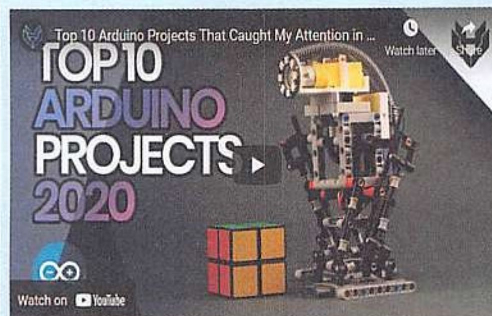
07 3D Printed Arduino Lawn Mower
http://bitly.ws/or9d

08 Web-Based Two-Player Game
http://bitly.ws/or9h

09 Control Robot Arm via Web
http://bitly.ws/or9o

10 Lightweight Arduino GSM Mobile Phone
http://bitly.ws/or9r

Top 10 Arduino Projects That Caught My Attention in 2020



▲ คำค้นหาใน YouTube : "Arduino Projects That Caught My Attention in 2020"

01 Zipline Robot
http://bitly.ws/or77

02 Contact-less Thermometer
http://bitly.ws/or6R

03 LEGO Elevator
http://bitly.ws/or7c

04 AI Learns to Play
http://bitly.ws/or7j

05 Arduino Hangman
http://bitly.ws/or7p

06 Thoughts And Prayers
http://bitly.ws/or7C

07 DIY Gimbal
http://bitly.ws/or8k

08 Pix-a-Sketch
http://bitly.ws/or7V

09 Biped Robot
http://bitly.ws/or86

10 Mimicking Robot Hand
http://bitly.ws/or8c

ศึกษา Arduino Projects ใน HOWTOMECHATRONICS

ที่เว็บไซต์ howtomechatronics.com คือแหล่งรวบรวม Arduino Projects ด้าน Mechatronics ดีๆ ให้ศึกษามากมาย พร้อมแสดงรายละเอียดการสร้างโครงงานแบบ Step by Step ไว้อีกด้วย โดยแต่ละโครงงานในเว็บนี้มีการสอนอย่างละเอียดที่เราสามารถลองทำตามได้เลย (DIY) พร้อมแจกจ่ายข้อมูลอย่างครบถ้วน



◀ เว็บไซต์รวม Arduino Projects ด้าน Mechatronics

องค์ประกอบการพัฒนา Arduino Projects

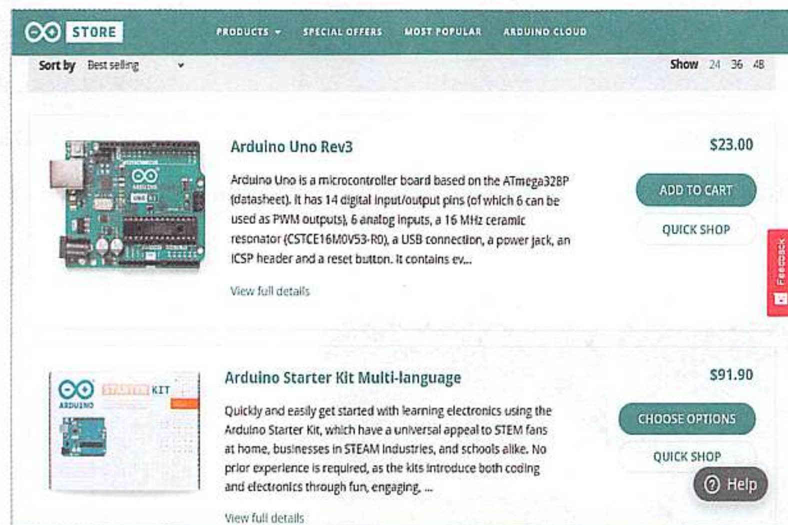
The word "Arduino" can mean 3 things

A Physical Piece of Hardware	A Programming Environment	A Community & Philosophy
		

▲ รูปแสดงองค์ประกอบการพัฒนา Arduino Projects

องค์ประกอบของการพัฒนา Arduino จะประกอบด้วย 3 ส่วนที่นำมาประยุกต์ใช้ทำงานร่วมกัน ได้แก่

1. A Physical Piece of Hardware คือ อุปกรณ์บอร์ด Arduino ที่ใช้ในการประมวลผลและควบคุมอุปกรณ์ Sensors ต่างๆ ด้วยไมโครคอนโทรลเลอร์ https://store.arduino.cc



▲ ภาพแสดงตัวอย่างบางส่วนของ Arduino Hardware

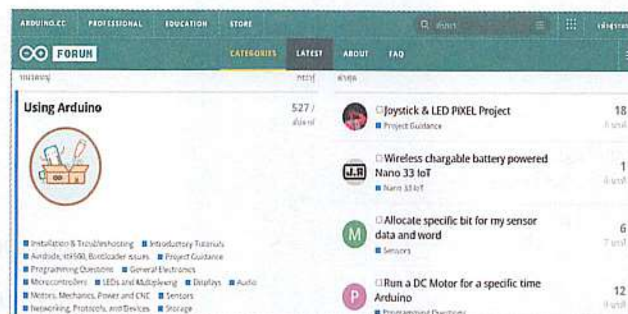
2. **A Programming Environment** คือ Arduino Software (IDE) สำหรับใช้เขียนโปรแกรมด้วยภาษา C เพื่อควบคุมสั่งการให้ Arduino ทำงานตามผู้พัฒนากำหนด โดยเลือกได้ 2 ทาง ได้แก่
   - **Online Tools** คือ การเขียนโปรแกรมออนไลน์บนหน้าเว็บไซต์ Arduino Web Editor
   - **Offline Tools** คือ การดาวน์โหลดและติดตั้งโปรแกรม Arduino IDE ลงในเครื่องคอมพิวเตอร์แล้วจึงเขียนโปรแกรม รองรับได้ทั้งระบบปฏิบัติการ Windows, Linux และ Mac OS X

ลิงค์ไปยังหน้าเว็บไซต์ Arduino Software (IDE) : www.arduino.cc/en/Main/Software



▲ รูปแสดงหน้าเว็บไซต์ Arduino Web Editor และหน้าต่างโปรแกรม Arduino IDE

3. **A Community & Philosophy** คือ แหล่งชุมชนความรู้ของนักพัฒนาโครงงาน Arduino ที่ใช้สอบถาม หาคำตอบข้อสงสัย แลกเปลี่ยนความรู้ซึ่งกันและกันผ่านทางฟอรัมหรือเว็บบอร์ด https://forum.arduino.cc



▲ รูปแสดง Arduino Forum ชุมชนออนไลน์



ซอฟต์แวร์เพื่อการเรียนรู้ Arduino

นอกจากซอฟต์แวร์ Arduino Software (IDE) แล้ว ยังมีซอฟต์แวร์ที่อยากแนะนำให้รู้จักที่จำเป็นต่อการใช้งานอีกหลายตัว เช่น ซอฟต์แวร์การเขียนโปรแกรมแบบ Block Code ที่ช่วยให้เขียนโปรแกรมได้โดยไม่จำเป็นต้องรู้โครงสร้างภาษา ที่เหมาะสำหรับเด็กประถม และซอฟต์แวร์จำลองการต่อวงจร หรือ Simulation Software ที่ช่วยให้เราลองต่อวงจร และดูผลลัพธ์แทนการต่อวงจรจริง ทำให้เราเรียนรู้ได้โดยไม่จำเป็นต้องมีอุปกรณ์จริง และซอฟต์แวร์สำหรับออกแบบวงจร หรือ Drawing Circuit เครื่องมือสำหรับออกแบบวงจรอิเล็กทรอนิกส์ ซึ่งจะแนะนำเพิ่มเติมในบทที่ 4

แหล่งข้อมูลศึกษาเพิ่มเติม (Arduino Tutorials)

แหล่งข้อมูลสำหรับผู้พัฒนาในการศึกษาเพิ่มเติมด้วยตัวเอง สามารถศึกษาได้ทาง Arduino Tutorials ประกอบด้วย 5 ส่วน ดังรูป



TUTORIALS ON ARDUINO PROJECT HUB

Arduino Project Hub is our official tutorial platform powered by hackster.io. Get inspired by a variety of tutorials, getting started guides, showcases and pro tips. Contribute projects and ideas, comment on the tutorials you are curious about, and 'Respect' the ones you like the most.



BUILT-IN EXAMPLES

Built-in Examples are sketches included in the Arduino Software (IDE), to open them click on the toolbar menu: File > Examples. These simple programs demonstrate all basic Arduino commands. They span from a Bare Minimum Sketch to Digital and Analog IO, to the use of Sensors and Displays.



EXAMPLES FROM LIBRARIES

The Arduino Software (IDE) can be extended through the use of libraries, just like most programming platforms, to provide extra functionality to your sketches. These tutorials walk you through the Examples of a number of libraries that come installed with the IDE, to open them click on the toolbar menu: File > Examples



FOUNDATIONS AND MORE

This section guides you though some of the key elements of the Arduino hardware and software, and the concepts behind them. What is a Sketch? What are Microcontrollers? What are the building blocks of the Arduino Programming language? Find these answers here.

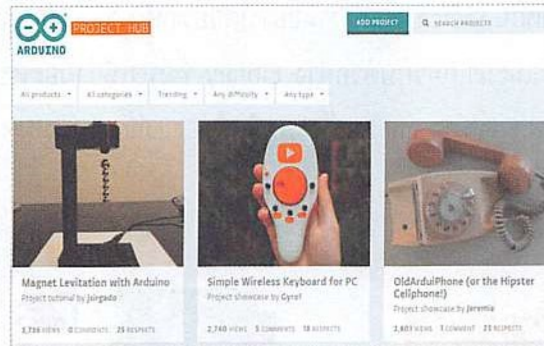


HACKING

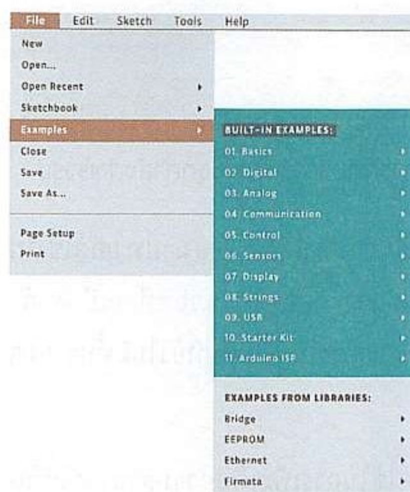
In this section you can find useful information to expand your knowledge about the Arduino platform. Do you want to know what's under the hood? Here you can find guidelines for customizing every software that runs on an Arduino board and the explanation of some of the hardware design details.

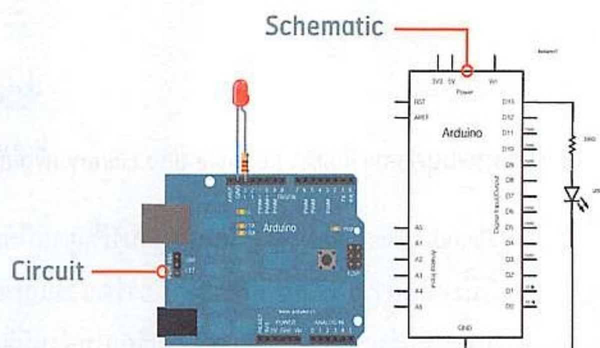
◀ รูปแสดงแหล่งศึกษาข้อมูลเพิ่มเติมด้านต่างๆ บนเว็บไซต์ Arduino www.arduino.cc/en/Tutorial/HomePage

1. **Tutorials On Arduino Project Hub :** เป็นแพลตฟอร์มแสดงผลงานที่สร้างด้วย Arduino เพื่อเป็นตัวอย่างให้เห็นสิ่งประดิษฐ์ เพื่อกระตุ้นไอเดียและสร้างแรงบันดาลใจในการประดิษฐ์ชิ้นงาน



2. **Built-In Examples :** เป็นตัวอย่าง Sketch ที่ให้มาพร้อมกับโปรแกรม Arduino Software IDE การเปิดใช้งานให้คลิกเมนู File > Examples บนแถบเครื่องมือ ก็จะแสดงการใช้งานคำสั่งพื้นฐานทั้งหมดรวม 11 เมนู ได้แก่ Basics, Digital, Analog, Communication, Control Structures, Sensors, Display, Strings, USB, Starter Kit & Basic Kit และ Arduino ISP อยากศึกษาหัวข้อไหนก็เลือกได้ตามความสนใจ ในตัวอย่างจะบอกหมดเลยว่า จะต้องเตรียมอุปกรณ์อะไรบ้าง (Hardware Requirement), ต่ออุปกรณ์กับบอร์ดอย่างไร (Circuit), แสดงผังวงจรไฟฟ้า (Schematic) และอธิบายคำสั่งที่เกี่ยวข้องพร้อมตัวอย่างชุดคำสั่ง (Code)

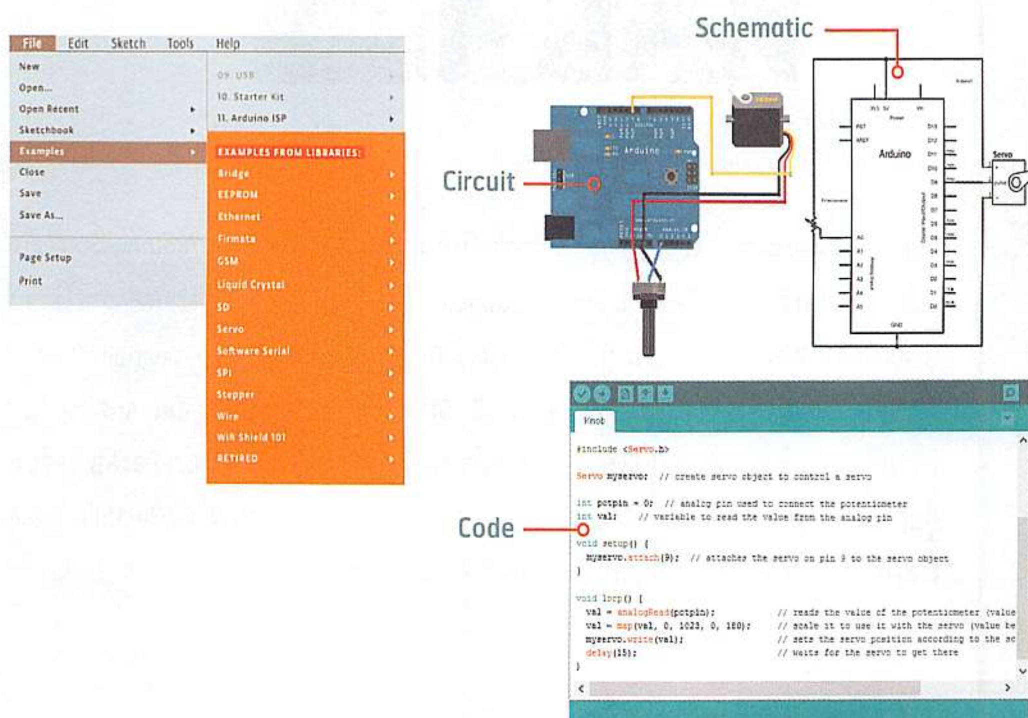






▲ รูปแสดงเมนูคำสั่งเพื่อเปิด Built-In Examples ของ Sketch เพื่อใช้ในการศึกษาการต่อวงจรและการเขียนโปรแกรม

3. **Examples from Libraries :** เป็นตัวอย่างแสดงการเรียกใช้ Library ในการเขียนโปรแกรมของ Arduino Software (IDE) ซึ่งก็จะเหมือนกับการเขียนโปรแกรมโดยทั่วไป เป็นการเรียกใช้ชุดคำสั่งสำเร็จรูปให้กับ Sketch เพื่อทำงานอย่างใดอย่างหนึ่ง วิธีนี้ช่วยให้เขียนโปรแกรมได้เร็วขึ้น ในบทเรียนนี้จะแนะนำเกี่ยวกับตัวอย่าง Library ที่มีมาให้ว่ามีอะไรบ้าง โดยเปิดดูตัวอย่าง Library ได้ที่เมนู File > Examples ก็จะพบลิสต์ของ Library ทั้งหมด



▲ รูปแสดงเมนูคำสั่งเพื่อเปิด Example ของ Library เพื่อใช้ในการศึกษาการเรียกใช้ชุดคำสั่งสำเร็จรูป

4. **Foundations and More :** เน้นให้ความรู้พื้นฐานที่ต้องมีในการจะนำ Arduino มาประยุกต์ใช้งาน มีทั้งหลักการและเทคนิคทางด้านฮาร์ดแวร์และซอฟต์แวร์ ตลอดจนแนวคิดที่อยู่เบื้องหลังแพลตฟอร์ม Arduino ทุกรุ่น บทเรียนในส่วนนี้จึงเหมาะสำหรับผู้เริ่มต้นที่ยังไม่มีพื้นฐานมาก่อนเลย

5. **Hacking :** เป็นแหล่งรวบรวมข้อมูลเชิงเทคนิค เพื่อใช้ในการเพิ่มเติมความสามารถให้กับบอร์ด Arduino ทั้งในด้านซอฟต์แวร์และฮาร์ดแวร์ ซึ่งจะเน้นให้ข้อมูลกับเมกเกอร์ (Maker) หรือนักประดิษฐ์ที่มีประสบการณ์แล้ว

Software :

- **Writing an Example :** คู่มือ Arduino สำหรับทุกๆ คนไม่ว่าจะเป็นมือสมัครเล่นหรือมืออาชีพก็ตาม ไม่ใช่เรื่องโครงสร้างการเขียนโปรแกรม และไม่ใช่ชุดกฎเกณฑ์ที่ตายตัว แต่เป็นแนวทางที่ช่วยให้โค้ดที่เราเขียนนั้นมีความชัดเจนสำหรับคนทุกระดับ
- **Writing a Library :** ข้อมูลการสร้าง Libraries เพื่อขยายความสามารถในการทำงานของ Arduino ทีละขั้นตอนตอนผ่านกระบวนการสร้าง Library จากการเขียน Sketch
- **Preferences :** ข้อมูลไฟล์การกำหนดค่าต่างๆ ของ Arduino ประกอบด้วยตัวเลือกมากมายสำหรับการปรับวิธีการ Compiles Arduino และการ Upload Sketch
- **Build Process :** ข้อมูลค้นหาขั้นตอนการ Build โปรแกรม Sketch ไปยังบอร์ด Arduino
- **Bootloader :** เป็นโปรแกรมขนาดเล็กที่จะรันตอน Arduino สตาร์ทการทำงาน หรือมีการกดปุ่ม Reset และช่วยในการอัพโหลดโปรแกรม Sketch ที่เราเขียนลงบนบอร์ด Arduino และยังมีเอกสารแนะนำ Bootload สำหรับบอร์ด Arduino Mini
- **Programmer :** ข้อมูลแสดงวิธีการเบิร์นโปรแกรม Sketch ที่เขียนลงบอร์ด Arduino ด้วยตัวโปรแกรมภายนอก เช่น AVR-ISP, STK500 หรือ Parallel Port เป็นต้น โดยไม่ต้องใช้ตัว Bootloader ทั้งนี้เพื่อประหยัดพื้นที่โปรแกรมบนชิปของ Arduino
- **Upgrading 8U2 Firmware :** ข้อมูลวิธีการอัพเกรดเฟิร์มแวร์ของชิป Atmega8U2 และ 16U2 บนบอร์ด Arduino Uno หรือ Arduino Mega2560 ซึ่งชิปนี้ทำหน้าที่เป็นตัวแปลง USB-to-Serial และสามารถอัพเดตผ่าน USB
- **Upgrading the WiFi Shield Firmware :** ให้ข้อมูลเกี่ยวกับวิธีการอัพเกรดเฟิร์มแวร์ของชิป ATmega32UC3A1256 ที่ควบคุมการส่ง TCP/IP Stack และสื่อสารกับโมดูล HDG104 WiFi การอัพเกรดเฟิร์มแวร์ก็เพื่อปรับปรุงการทำงานของ WiFi Shield
- **Upgrading the 16U2 Firmware on the Due :** ให้ข้อมูลเกี่ยวกับวิธีการอัพเกรดเฟิร์มแวร์ของชิป Atmega16U2 บนบอร์ด Arduino Due ซึ่งทำหน้าที่เป็นตัวแปลง USB-to-Serial บน Programming Port
- **Source :** แหล่งรวม Source Code ของ Arduino
- **Bugs :** แสดง Bug ที่พบในปัจจุบันของซอฟต์แวร์ Arduino และการปรับปรุงใหม่

Hardware :

- **Pin Mapping (ATmega8, ATmega168) :** ข้อมูลแผนภาพแสดงการติดต่อของขา Pin บนบอร์ด Arduino ที่ใช้ชิปไมโครคอนโทรลเลอร์ ATmega8 หรือ ATmega168
- **SERCOM - Adding more Serial Interfaces to SAMD Microcontrollers :** แนะนำการสร้าง Serial Ports ด้วยโปรโตคอล I2C, SPI หรือ UART บนบอร์ด SAMD
- **NG Auto Reset :** วิธีแก้ไข Arduino NG เพื่อให้สามารถ Upload Sketches โดยไม่ต้องกดปุ่มรีเซตบนบอร์ด
- **Parallel Programmer :** คำแนะนำในการสร้างฮาร์ดแวร์ราคาถูกที่ช่วยให้เบิร์น Bootloader ไปยังบอร์ด Arduino หรืออัพโหลดโปรแกรม Sketch ไปยังบอร์ดที่ไม่มี Bootloader (ใช้งานได้กับคอมพิวเตอร์ที่มีพอร์ตขนานเท่านั้น)
- **Breadboard Circuit :** แนะนำการจำลองวงจรของบอร์ด Arduino บน Breadboard

เทคนิคการปรับแต่ง Arduino :

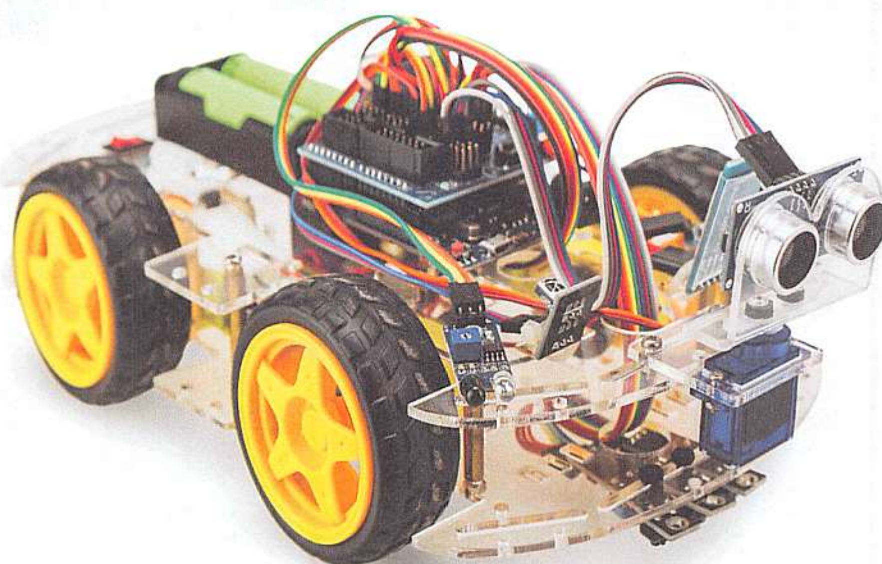
- **DIY Breadboard Shield :** ตัวอย่างการทำ Breadboard Shield พร้อมการติดตั้งเข้ากับบอร์ด Arduino
- **DIY Arduino Shields :** แนะนำการออกแบบและกัดแผ่นพรินต์ Arduino Shield ด้วยตนเอง

แหล่งข้อมูลอื่นๆ :

- **AVR Fuse Calculator :** การตั้งค่า Fuse ให้เหมาะกับวงจร
- **Minty Boost Process :** อธิบายวิธีการประกอบชุด Kit เข้าด้วยกัน การนำไอเดียไปสู่การสร้างเป็นผลิตภัณฑ์ ++

บทสรุปท้ายบท

สำหรับเนื้อหาในบทแรกนี้ ได้พาทุกท่านไปรู้จักกับโลกของ Arduino ในภาพรวม ให้รู้ว่ามันคืออะไร นำไปใช้ประโยชน์อย่างไร จุดกำเนิดความเป็นมา การใช้ Arduino เพื่อการศึกษา แนะนำผลิตภัณฑ์และการเลือกใช้ องค์ประกอบของการพัฒนา Arduino Projects แหล่งเรียนรู้เพิ่มเติมในรูปของคลิป VDO ที่แสดงตัวอย่างโครงงานมากมายจากทั่วโลก เพื่อให้ผู้เริ่มต้นได้เห็นภาพรวมทั้งหมดก่อนที่จะเริ่มศึกษาลึกลงไปที่เรื่องของฮาร์ดแวร์ ซึ่งก็คือ บอร์ดและอุปกรณ์เสริมต่างๆ และศึกษาด้านซอฟต์แวร์การเขียนโปรแกรมควบคุม ซึ่งต้องเข้าใจโครงสร้างภาษา เมื่อมีพื้นฐานความรู้จากทั้ง 2 ส่วนแล้ว ต่อไปก็จะเป็นภาคปฏิบัติที่ต้องลงมือทำ LAB เพื่อเรียนรู้การทำงาน แต่ในเบื้องต้นจะให้ศึกษากับซอฟต์แวร์จำลองก่อน คือ การต่อวงจรโดยยังไม่ต้องใช้อุปกรณ์จริงนั่นเอง เมื่อเราพอจะเข้าใจเรื่องของวงจรดีแล้ว จะซื้ออุปกรณ์จริงมาประกอบเป็นชิ้นงานก็สามารถทำได้เลย

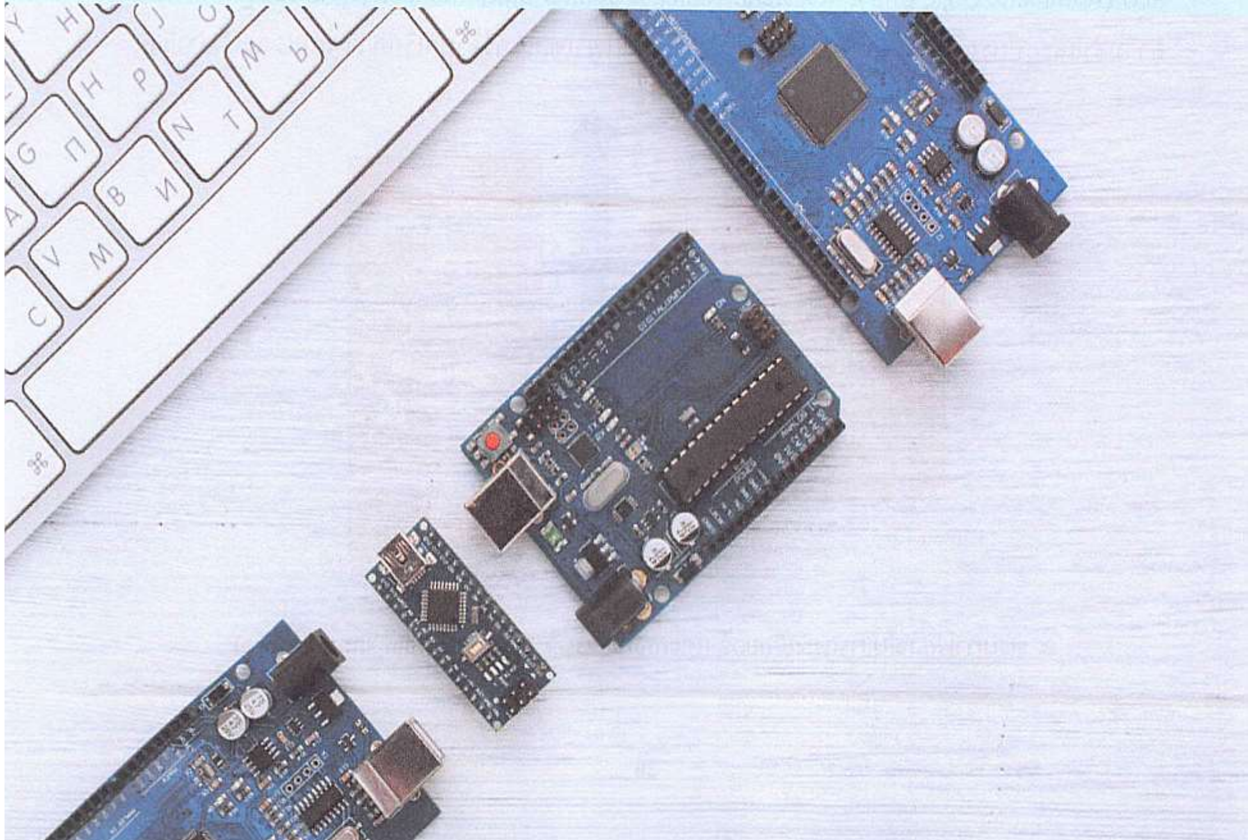


CHAPTER

02

รู้จักบอร์ดมาตรฐานของ Arduino และอุปกรณ์เสริม

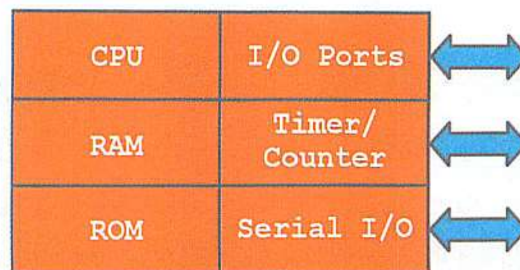
เราได้รู้จักกับ Arduino ในภาพรวมกันไปแล้วในบทแรก สำหรับเนื้อหาในบทที่ 2 นี้ เราจะลงลึกในรายละเอียดของอุปกรณ์มากขึ้น เพื่อดูว่าบอร์ดไมโครคอนโทรลเลอร์มีส่วนประกอบอะไรบ้าง ส่วนไหนมีหน้าที่ทำอะไร โดยจะเริ่มที่ Arduino UNO ที่ถือเป็นบอร์ดรุ่นมาตรฐานที่เหมาะสำหรับผู้เริ่มต้นมากที่สุด หลังจากนั้นจะแนะนำบอร์ดตระกูลอื่นๆ ที่ได้รับความนิยม (Most Popular) พร้อมแนะนำอุปกรณ์เสริมอื่นๆ และตัวอย่างการประยุกต์ใช้งานบอร์ด Arduino กับ Sensors



โครงสร้างของ Microcontroller

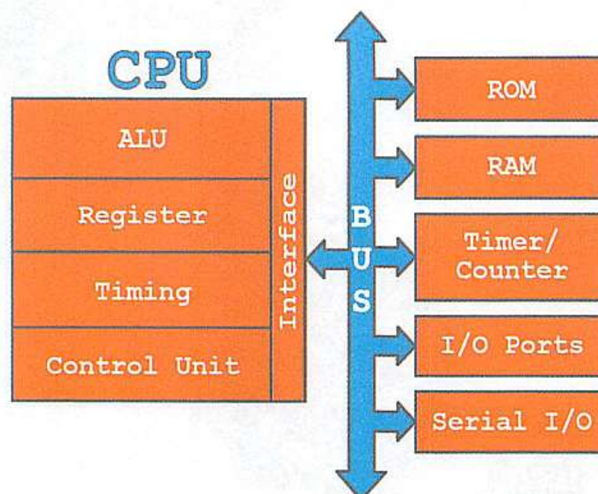
ในบทแรกเราได้กล่าวไปแล้วว่า ไมโครคอนโทรลเลอร์คืออะไร ในบทที่สองนี้จะแสดงโครงสร้างโดยทั่วไปของบอร์ดไมโครคอนโทรลเลอร์ว่ามีส่วนประกอบอะไรบ้าง

ไมโครคอนโทรลเลอร์ (Microcontroller) จะประกอบด้วย CPU, RAM, ROM, I/O Ports, Timer/Counter และ Serial I/O ทั้งหมดที่กล่าวมานี้จะถูกบรรจุไว้ในชิปตัวเดียวกัน เรียกว่า **"Computer-on-a-Chip"**

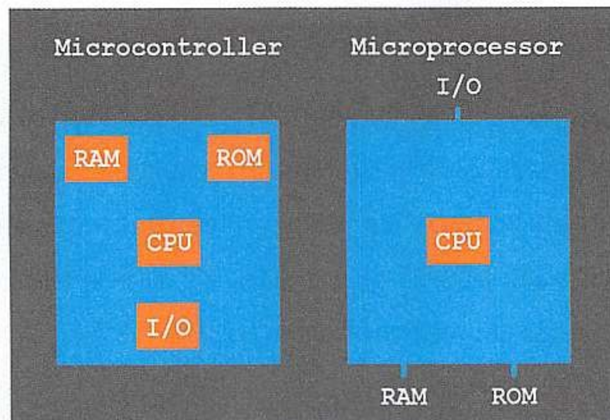


▲ แผนภาพของไมโครคอนโทรลเลอร์ (Microcontroller Block Diagram)

ไมโครโพรเซสเซอร์ (Microprocessor) ประกอบด้วย CPU (Central Processing Unit) ซึ่งมี ALU (Arithmetic Logic Unit), Registers, Timing & Control Unit, Interface (ส่วนเชื่อมต่อกับหน่วยความจำและอุปกรณ์ I/O) ใส่ไว้บนชิปตัวเดียว ไมโครโพรเซสเซอร์จึงถูกเรียกว่า **"CPU-on-a-Chip"**



▲ แผนภาพของไมโครโพรเซสเซอร์ (Microprocessor Block Diagram และ Interface)



▲ Microprocessor ภายในมีเพียง CPU เท่านั้น ไม่มี Memory ซัพพอร์ต ส่วน Microcontroller มีทั้ง CPU, RAM, ROM และ I/O Ports ฝังอยู่ภายในชิป จึงเปรียบได้กับระบบคอมพิวเตอร์ขนาดเล็ก

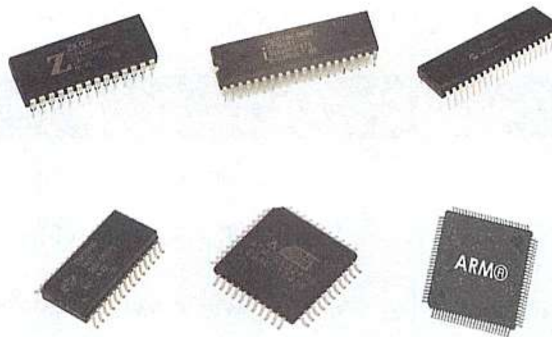
เปรียบเทียบ Microprocessor & Microcontroller

Microprocessor	Microcontroller
CPU-on-a-Chip	Computer-on-a-Chip
ประยุกต์ใช้ในงานหลากหลาย	มีหน้าที่ชัดเจนว่าต้องทำงานอะไร
ถูกใช้ในระบบคอมพิวเตอร์ทั่วไป	ถูกใช้ในระบบควบคุมฝังตัว/สมองกลฝังตัว (Embedded Systems)
ส่วนของ Memory และ I/O เป็นอุปกรณ์เชื่อมต่อภายนอก (อยู่นอกชิป)	ส่วนของ Memory และ I/O ถูกติดตั้งมาแล้วภายใน (ติดตั้งภายในชิป)
ส่วนประกอบทั้งหมดอยู่ภายนอกชิป จึงต้องติดตั้งผ่าน Interface ทำให้ทั้งระบบมีขนาดใหญ่และแพงกว่า	ส่วนประกอบทั้งหมดอยู่ภายในชิป ทำให้ระบบเล็กกว่าและถูกกว่า
ในระบบที่ใช้ไมโครโพรเซสเซอร์ ชุดคำสั่งและข้อมูลจะถูกเก็บไว้ในโมดูลหน่วยความจำเดียวกัน	ในระบบที่ใช้ไมโครคอนโทรลเลอร์หน่วยความจำของโปรแกรม และหน่วยความจำของข้อมูลจะถูกเก็บแยกกัน

ตระกูลของบอร์ดและชิป Microcontroller

หากย้อนเวลากลับไป 20 กว่าปีก่อน คนที่เคยเรียนวิชาไมโครโพรเซสเซอร์ในสายอิเล็กทรอนิกส์ก็น่าจะเคยสัมผัสกับบอร์ดที่ใช้ชิป Z80 ซึ่งถือเป็นชิปไมโครโพรเซสเซอร์ ต่อมาก็นิยมเรียนบอร์ดที่ใช้ชิปไมโครคอนโทรลเลอร์กันมากขึ้นในสถานศึกษาต่างๆ โดยชิปไมโครคอนโทรลเลอร์นั้นมีอยู่หลายตระกูลจากผู้ผลิตหลายค่าย เช่น MCS-51 ของค่าย Intel, PIC ของค่าย Microchip, AVR ของค่าย Atmel, Psoc ของค่าย Cypress และ ARM ของค่าย Arm Holdings

พอมาถึงยุคนี้ บอร์ดไมโครคอนโทรลเลอร์ที่ได้รับความนิยมก็ต้องยกให้กับบอร์ด Arduino ที่ใช้ AVR Microcontrollers ของค่าย Atmel และบอร์ด Raspberry Pi ก็ใช้ AVR Microcontroller ของค่าย Atmel เช่นเดียวกัน



▲ รูปแสดงตัวอย่างชิป Microcontroller จากอดีตถึงปัจจุบัน

รู้จักส่วนประกอบของบอร์ด Arduino

บอร์ด Arduino มีมากมายหลายรุ่นที่เปิดตัวในตลาด เช่น Arduino UNO, Arduino DUE, Arduino LEONARDO, Arduino MEGA แต่รุ่นที่พบบ่อยที่สุด คือ Arduino UNO และ Arduino MEGA หากใครวางแผนที่จะสร้างโครงงานที่เกี่ยวข้องกับอุปกรณ์อิเล็กทรอนิกส์ระบบฝังตัวหุ่นยนต์ หรือ IoT การใช้ Arduino UNO จะเป็นตัวเลือกที่ง่ายและประหยัดที่สุด ในหัวข้อนี้จะแนะนำให้รู้จักกับบอร์ด Arduino ทั้ง 4 รุ่นที่กล่าวมา

บอร์ด Arduino UNO

ถ้านี่คือครั้งแรกที่ต้องข้องเกี่ยวกับอุปกรณ์ทางอิเล็กทรอนิกส์และการเขียนโปรแกรม บอร์ด UNO เป็นรุ่นที่เหมาะสมที่สุด เพราะบอร์ดรุ่นนี้ถูกออกแบบมาอย่างเรียบง่ายที่สุด ตัดเอาความซับซ้อนที่ยังไม่จำเป็นสำหรับมือใหม่ออกไป เพื่อให้คนส่วนใหญ่ไม่ว่าจะเป็นรุ่นเล็กหรือรุ่นใหญ่ จะมีพื้นฐานการศึกษาทางสายอิเล็กทรอนิกส์หรือไม่ก็ตาม สามารถเรียนรู้ได้ไม่ยาก บอร์ดรุ่น UNO จัดว่าได้รับความนิยมมากที่สุดในตระกูลของ Arduino

บอร์ด Arduino UNO มันมีทุกสิ่งที่จำเป็นสำหรับระบบควบคุมขนาดเล็ก จัดเป็นบอร์ดไมโครคอนโทรลเลอร์ระดับ Basic หรือที่เรียกว่า "รุ่นมาตรฐาน" เวลาจะใช้งานก็เพียงแค่ต่อสาย USB เชื่อมระหว่างบอร์ดกับคอมพิวเตอร์ หรือจะเลือกจ่ายพลังงานจากอะแด็ปเตอร์ AC-to-DC หรือแบตเตอรี่ก็ได้ เมื่อจ่ายไฟเข้าที่บอร์ดแล้วมันก็จะพร้อมทำงานทันที

บอร์ด Arduino UNO ถูกออกแบบมาอย่างรู้ใจมือใหม่ที่ยังขาดประสบการณ์ ลองนึกภาพว่าหากพลาดพลั้งต่อวงจรผิดจนเป็นเหตุให้ชิปไหม้ อะไรจะเกิดขึ้น และถ้าเกิดเหตุการณ์นี้ขึ้นมาจริงๆ เราก็แค่ไปซื้อชิปมาเปลี่ยนใหม่เองได้เลย เพราะชิปจะติดตั้งอยู่บน Socket ที่ถอดเปลี่ยนเองได้ง่ายๆ

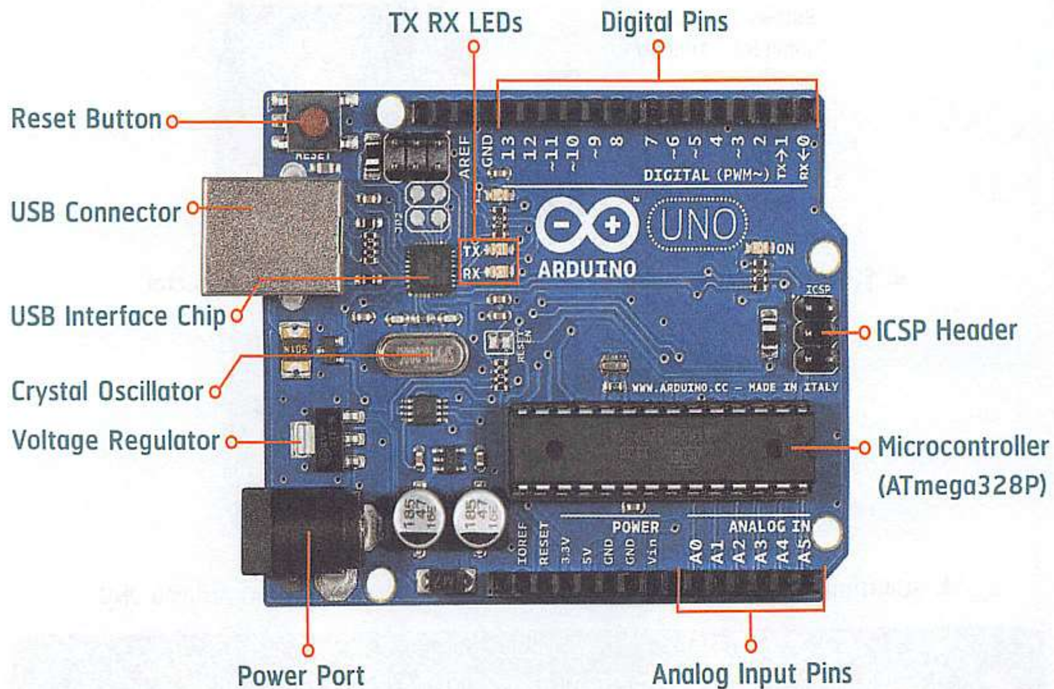


▲ รูปแสดงบอร์ด Arduino UNO Revision 3 ในมุมมองต่างๆ

Arduino UNO Revision 3 ถูกพัฒนามาอย่างต่อเนื่องจนถึงรุ่นที่ 3 แล้ว เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ชิปรุ่น ATmega328P (Datasheet) มีขารับและขาส่งสัญญาณดิจิทัล 14 ขา (มีอยู่ 6 ขา ที่ใช้เป็นเอาต์พุต PWM), ขารับสัญญาณอะนาล็อก 6 ขา, คริสตัลให้กำเนิดสัญญาณความถี่หรือสัญญาณนาฬิกา (Clock Speed) 16 MHz, พอร์ตสำหรับต่อสาย USB (เพื่อเขียนโปรแกรมหรืออัพโหลด Sketch และป้อนไฟเลี้ยงในตัว), แจ็คต่อไฟเลี้ยง, หัว ICSP และปุ่ม Reset

ตารางแสดง TECH SPECS	
Microcontroller (MCU)	ATmega328P
Operating Voltage	5 V
Input Voltage (Recommended)	7-12 V
Input Voltage (Limit)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Reference : https://store.arduino.cc/usa/arduino-uno-rev3



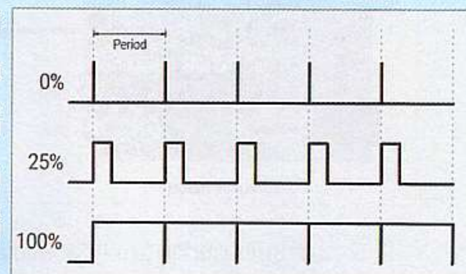
▲ รูปแสดงส่วนประกอบของบอร์ด Arduino UNO Rev 3



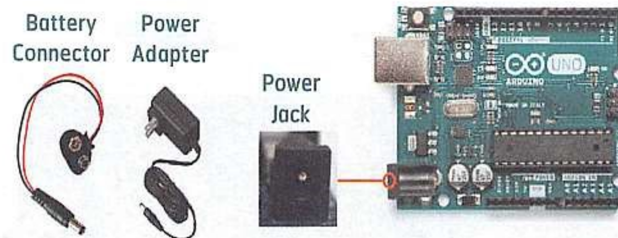
PWM คืออะไร?

PWM ย่อมาจากคำว่า "Pulse Width Modulation" คือ เทคนิคการเปลี่ยนช่วงความถี่ให้เหมาะสมกับการส่งสัญญาณ หรือที่เรียกว่า "Modulation" ถูกใช้เพื่อเข้ารหัสข้อความแปลงเป็นสัญญาณพัลส์ (Pulse) ซึ่งเป็นสัญญาณไฟฟ้าที่มีลักษณะเป็นคลื่นรูปสี่เหลี่ยม มีเพียงสถานะ High หรือ Low ต่อเนื่องซ้ำกันไปเรื่อยๆ ช่วงเวลาที่สัญญาณมีสถานะเป็น High ในหนึ่งรอบ (Period) นั้นจะถูกเรียกว่า "ความกว้างพัลส์" (Pulse Width) ซึ่งนิยมแสดงเป็นเปอร์เซ็นต์ ดังรูป

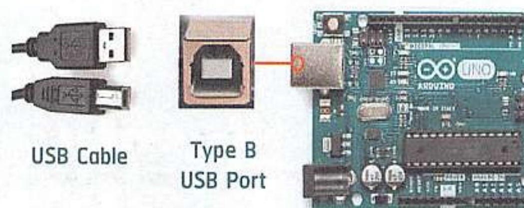
PWM เป็นเทคนิคที่ Arduino ใช้ในการควบคุมวงจร เช่นใช้ควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor), ควบคุมการหรี่แสงของหลอดแอลอีดี (Dimming LED) เป็นต้น เมื่อขา (Pin) บนบอร์ดถูก Enable เพื่อเปิดใช้งาน PWM มันจะสร้างความถี่คงที่ ~ 500 Hz ในขณะที่รอบการทำงานจะเปลี่ยนไปตามพารามิเตอร์ที่ผู้ใช้ตั้งค่า



▲ รูปแสดงเปอร์เซ็นต์ของสัญญาณ Pulse



▲ รูปแสดง Power Jack ที่ใช้ต่อกับ Power Adapter หรือ Battery Connector

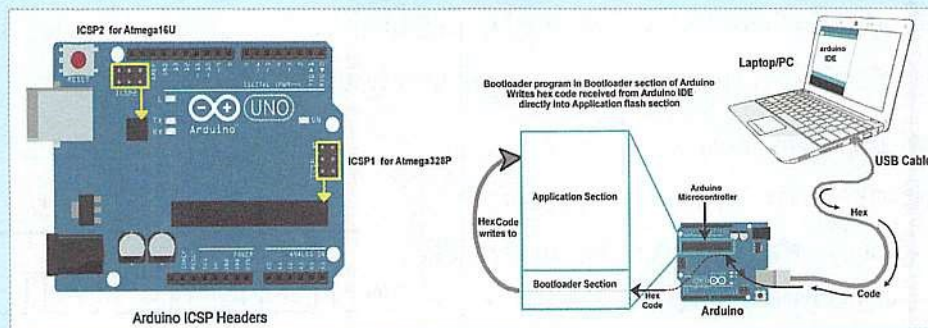


▲ รูปแสดงพอร์ต USB Type B สำหรับเชื่อมต่อกับคอมพิวเตอร์กับบอร์ด Arduino UNO



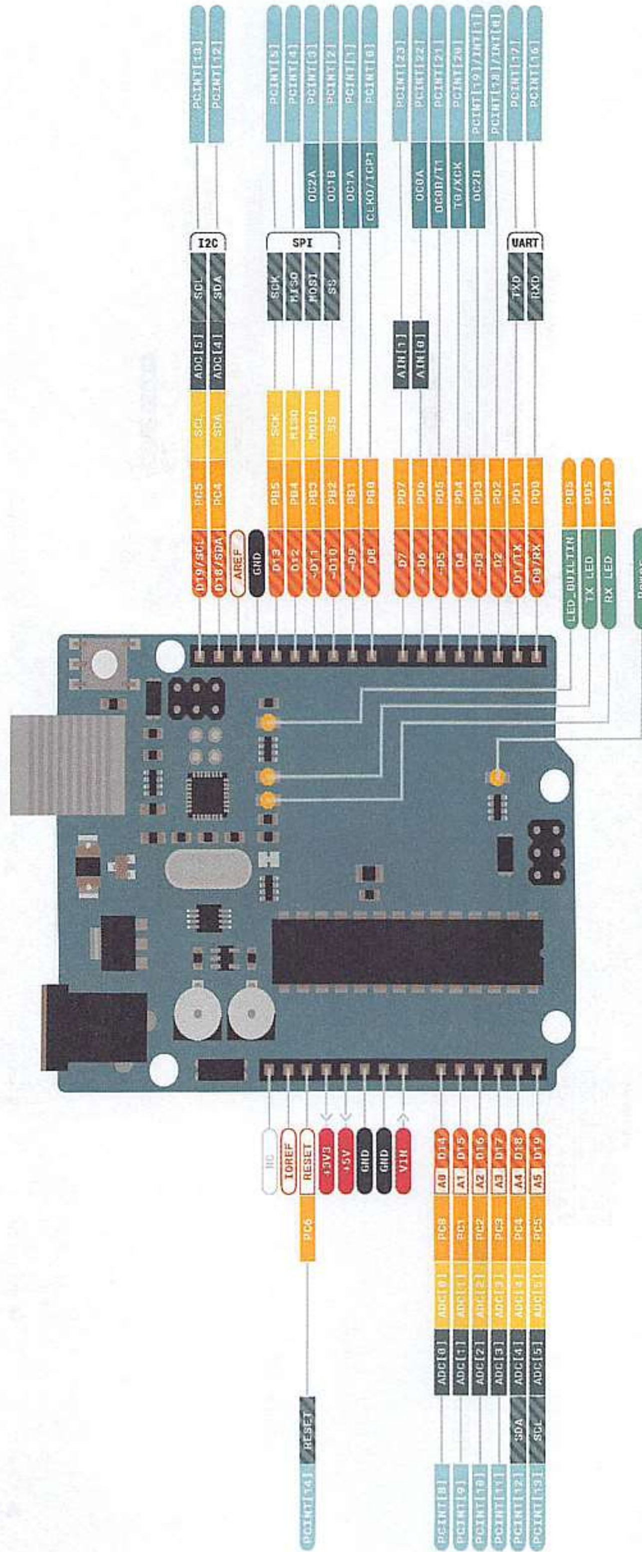
วิธีพัฒนาโค้ดเริ่มต้นการทำงานของบอร์ด (Flash Arduino Bootloader)

ในการปรับปรุงโค้ดเริ่มต้นการทำงานของบอร์ด (Flash Bootloader) ใหม่ เราต้องเขียนหรือติดตั้ง Bootloader ลงใน Bootloader Section ในพื้นที่หน่วยความจำ โดยปกติแล้วเราจะเขียนมันลงในชิป IC ก่อนจะบัดกรีลงบนแผ่น PCB ขณะที่ผู้ผลิตชิปไมโครคอนโทรลเลอร์บางราย เช่น Atmel หรือ Microchip ได้เตรียม ICSP Header (In-Circuit Serial Programming) ไว้ให้บนบอร์ดเพื่อใช้สำหรับแฟลชเมมโมรี สำหรับบอร์ด Arduino UNO จะมี ICSP Headers ไว้ให้ 2 หัว หัวหนึ่งสำหรับแฟลชชิป ATmega16U2 ที่ตำแหน่ง ICSP2 ส่วนอีกหัวสำหรับแฟลชชิป ATmega328 ที่ตำแหน่ง ICSP1 ดังรูป



▲ รูปแสดงตำแหน่ง ICSP Header บนบอร์ด Arduino UNO และวิธแฟลช

www.electronicwings.com/arduino/basics-to-developing-bootloader-for-arduino

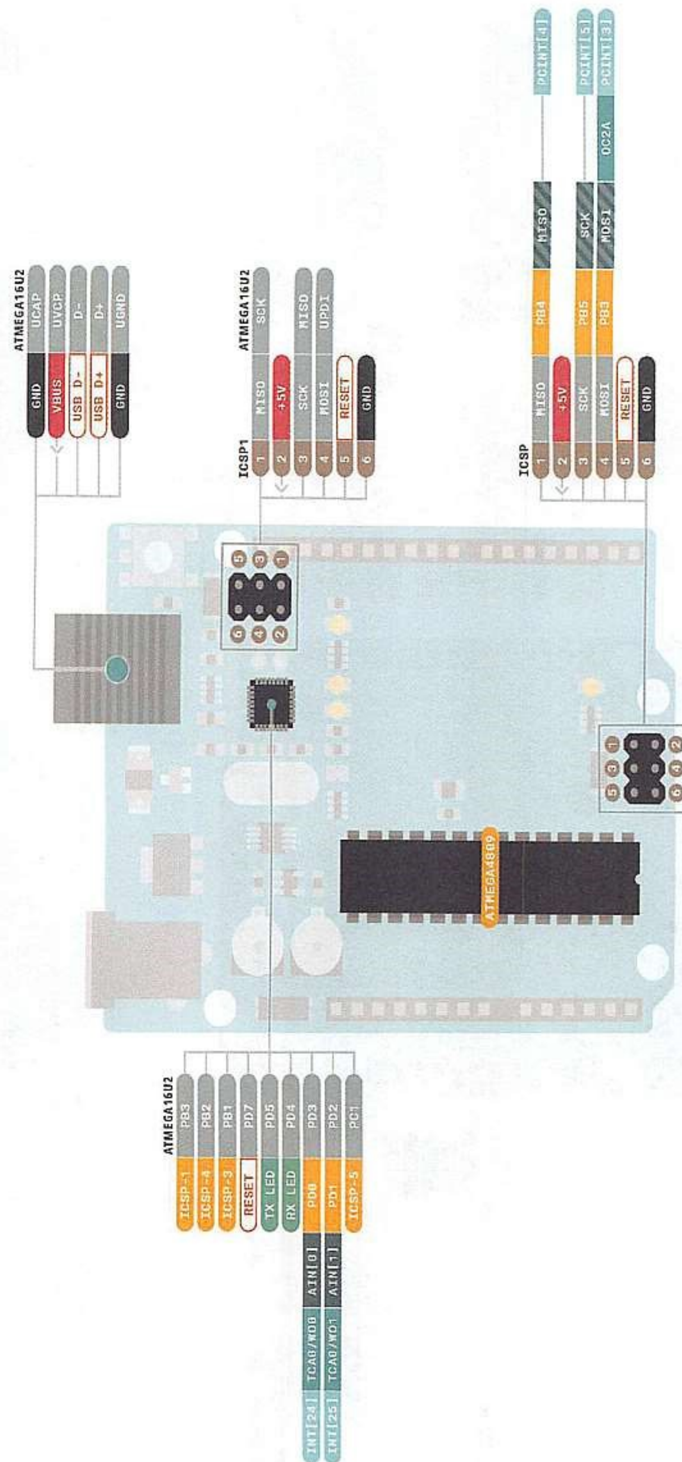




▲ รูปแสดงส่วนประกอบของบอร์ด Arduino UNO Pinout – Diagram

ARDUINO UNO REV3

STORE.ARDUINO.CC/UNO-REV3







Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom

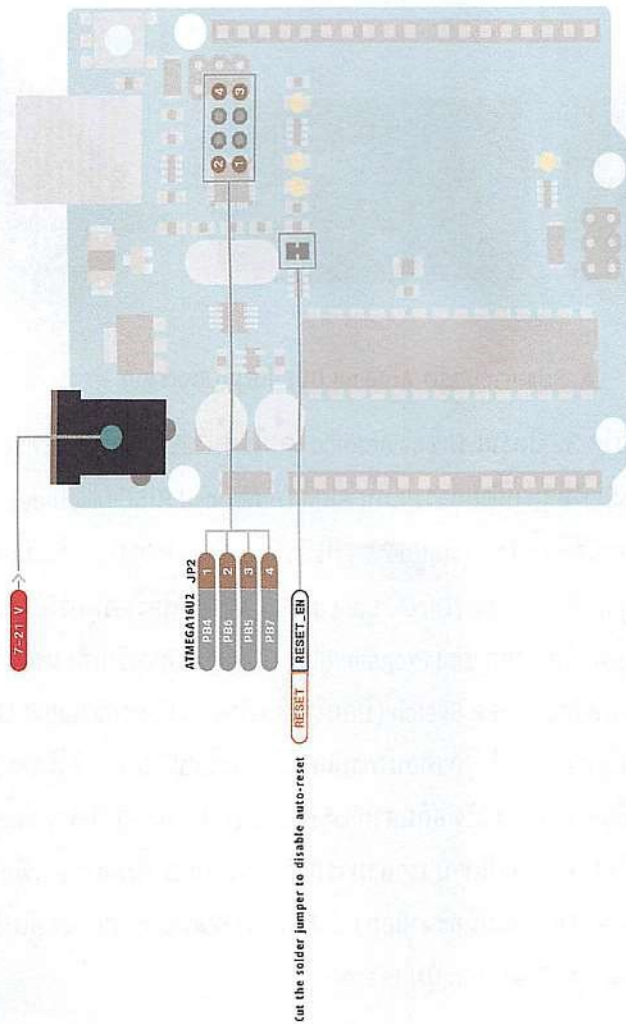
MAXIMUM current per I/O pin is 20mA

MAXIMUM current for +3.3V pin is 50mA

VIN 6-20 V input to the board.



ARDUINO
UNO REV3
STORE.ARDUINO.CC/UNO-REV3





- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default

SJ Pin: Making a short circuit using the solder jumper allows only the function in the SJ Pin cells.

MAXIMUM current per I/O pin is 20mA

MAXIMUM current for +3.3V pin is 50mA

VIN 6-20 V input to the board.

ARDUINO.CC

บอร์ด Arduino DUE

Arduino DUE เป็นบอร์ดขนาด 32 บิตตัวแรก โดยผู้ผลิต Arduino.cc ได้พัฒนารุ่นนี้ด้วยความตั้งใจที่จะสร้างบอร์ดสำหรับผู้เริ่มต้น ที่ใช้งานง่ายแม้ไม่มีความรู้ทางเทคนิคมาก่อน แค่เสียบบอร์ดเข้ากับคอมพิวเตอร์ผ่านสาย USB ก็เริ่มทำงานทันที จุดเด่นของบอร์ด Arduino DUE คือ มีพลังในการประมวลผลที่รวดเร็ว แต่ข้อเสียของบอร์ด Arduino DUE คือ มีขนาดค่อนข้างใหญ่ จึงอาจไม่เหมาะกับสิ่งประดิษฐ์หรือโครงงานที่มีพื้นที่จำกัด นับเป็นบอร์ดที่มีสมรรถนะสูง เหมาะที่จะนำไปใช้ในโปรเจ็กต์ขนาดใหญ่

Arduino DUE เป็นโมดูลขนาดใหญ่เมื่อเปรียบเทียบกับ Arduino UNO โดยมีจำนวน Pins และพื้นที่หน่วยความจำที่มากกว่าเมื่อเปรียบเทียบกัน



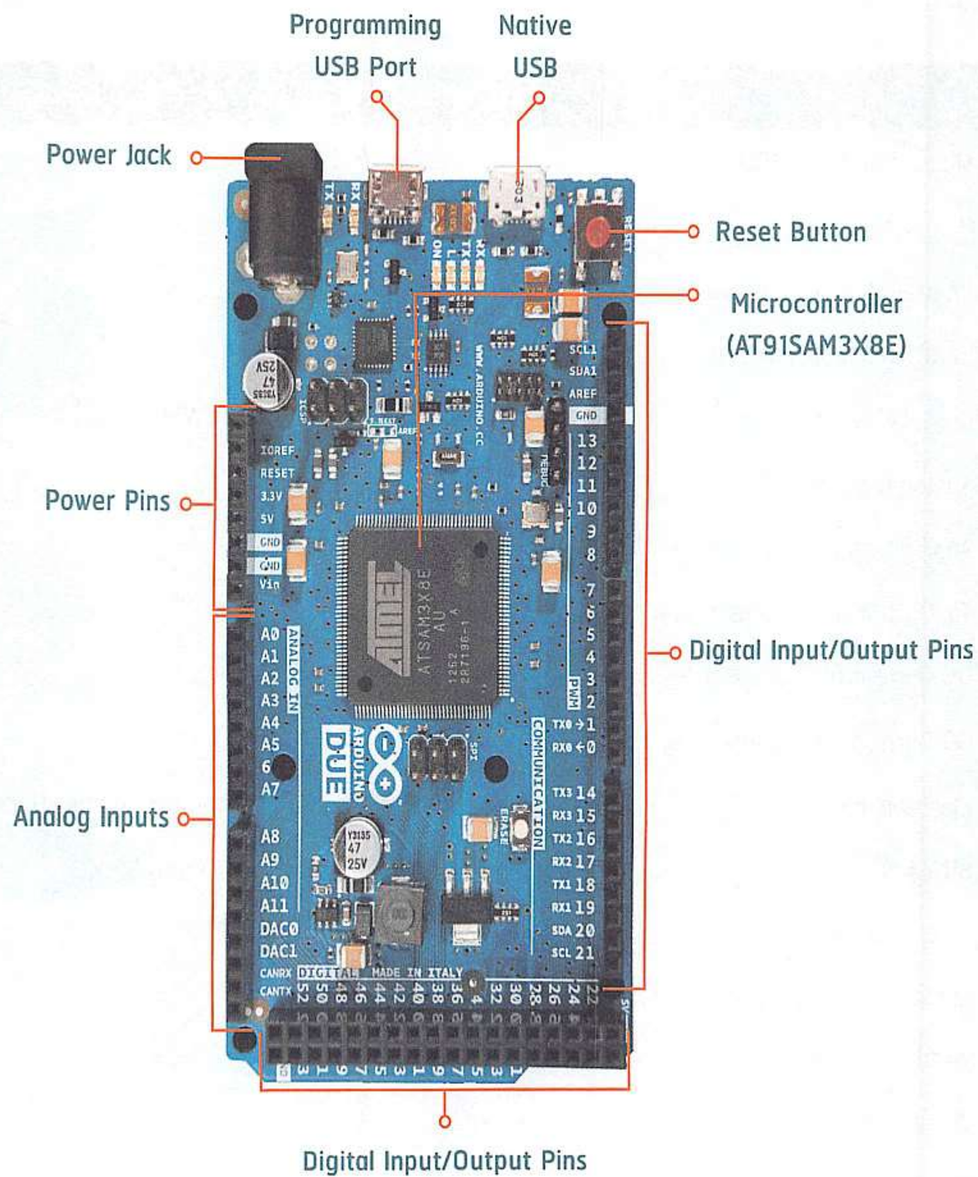
▲ รูปแสดงบอร์ด Arduino DUE ในมุมมองต่างๆ

Arduino DUE เป็นบอร์ด 32 บิตรุ่นแรกของ Arduino โดยใช้ชิปไอซีไมโครคอนโทรลเลอร์รุ่น Atmel SAM3X8E ARM Cortex-M3 ที่มีสถาปัตยกรรมแบบ ARM (Advanced RISC Machine) บอร์ดประกอบด้วยขารับและขาส่งสัญญาณดิจิทัล 54 ขา (มีอยู่ 12 ขา ที่ใช้เป็นเอาต์พุต PWM), ขารับสัญญาณอะนาล็อก 12 ขา, คริสตัลผลิตสัญญาณที่ความเร็ว (Clock Speed) 84 MHz, พอร์ตอนุกรม (Serial Ports) 4 ช่อง, บอร์ดมี USB มาให้ 2 พอร์ต ได้แก่ ช่อง Programming USB Port สำหรับเชื่อมต่อกับคอมพิวเตอร์เพื่อใช้ในการเขียนโปรแกรมหรืออัพโหลด Sketch (ป้อนไฟเลี้ยงในตัว) และช่อง Native USB Port พอร์ตที่มีการผนวกวงจรสื่อสารไว้ภายในชิปไมโครคอนโทรลเลอร์ มีประโยชน์ในการใช้ไมโครคอนโทรลเลอร์เพื่อจำลองอุปกรณ์ USB ที่แตกต่างกัน ใช้ต่อกับสาย USB OTG (On-The-Go) สำหรับอุปกรณ์ต่อพ่วง เช่น คีย์บอร์ดและสมาร์ทโฟน อย่างไรก็ตาม เราสามารถใช้พอร์ต USB ทั้งสองช่องเพื่อใช้ในการเขียนโปรแกรมได้เหมือนกัน, 2 DAC (ดิจิทัลเป็นอะนาล็อก), 2 TWI (ขา SDA และ SCL จะอยู่ใกล้กับขา AREF), แจ็คไฟ, หัว SPI, หัว JTAG, ปุ่ม Reset และปุ่ม Erase

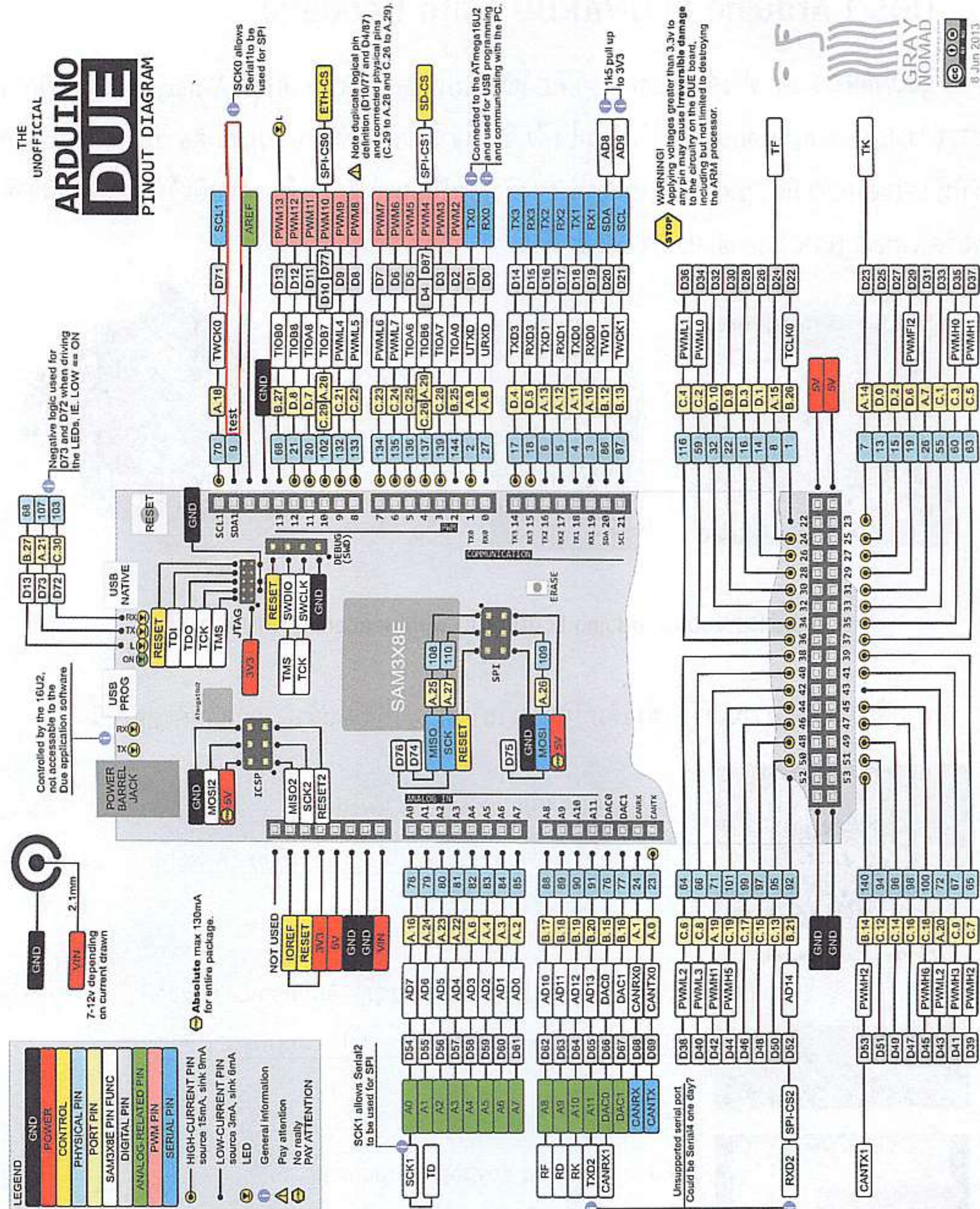
สรุปว่าบอร์ด DUE ประกอบด้วยทุกสิ่งที่จำเป็นเพื่อรองรับไมโครคอนโทรลเลอร์ เพียงเชื่อมต่อกับคอมพิวเตอร์ด้วยสาย Micro-USB หรือจะจ่ายไฟผ่านอะแด็ปเตอร์ AC-to-DC หรือใช้แบตเตอรี่ก็เป็นอีกทางเลือกหนึ่ง

ตารางแสดง TECH SPECS	
Microcontroller (MCU)	AT91SAM3X8E
Operating Voltage	3.3 V
Input Voltage (Recommended)	7-12 V
Input Voltage (Limits)	6-16 V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O Lines	130 mA
DC Current for 3.3 V Pin	800 mA
DC Current for 5 V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks : 64 KB and 32 KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Reference : https://store.arduino.cc/usa/due



▲ รูปแสดงส่วนประกอบของบอร์ด Arduino DUE

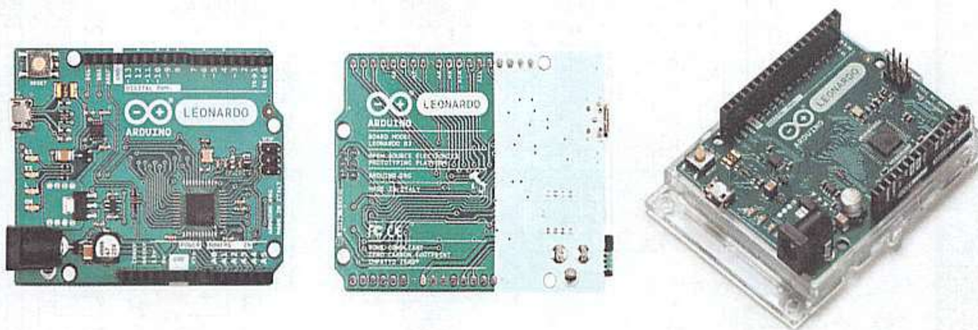


▲ รูปแสดงส่วนประกอบของบอร์ด Arduino DUE Pinout - Diagram

Reference : https://forum.arduino.cc/index.php?topic=132130.0

บอร์ด Arduino LEONARDO (with Headers)

LEONARDO แตกต่างจากบอร์ดอื่นๆ คือ ใช้ชิปไมโครคอนโทรลเลอร์รุ่น ATmega32u4 ซึ่งมีการบรรจุ USB Communication เอาไว้พร้อมในตัว จึงไม่จำเป็นต้องใช้ตัวประมวลผลสำรองจากภายนอก ทำให้ LEONARDO มีความสามารถในการจำลองตัวเองเป็นเมาส์ คีย์บอร์ด จอยสติกได้ นอกจากนี้ยังมีพอร์ต Virtual (CDC) Serial Port/COM อีกด้วย



▲ รูปแสดงบอร์ด Arduino LEONARDO with Headers ในมุมมองต่างๆ

หากต้องการทำความเข้าใจเพิ่มเติม ให้เปิดเว็บไซต์ YouTube แล้วค้นหาคลิปดังต่อไปนี้



คลิป : "DIY Arduino Mouse Joystick"

https://www.youtube.com/watch?v=t8mE1ayw5bo



คลิป : "Joystick as mouse using Arduino Leonardo"

https://www.youtube.com/watch?v=u0DYwo009Yo



คลิป : "Arduino Keyboard Emulator"

https://www.youtube.com/watch?v=SHIcliL4O14&t=3s

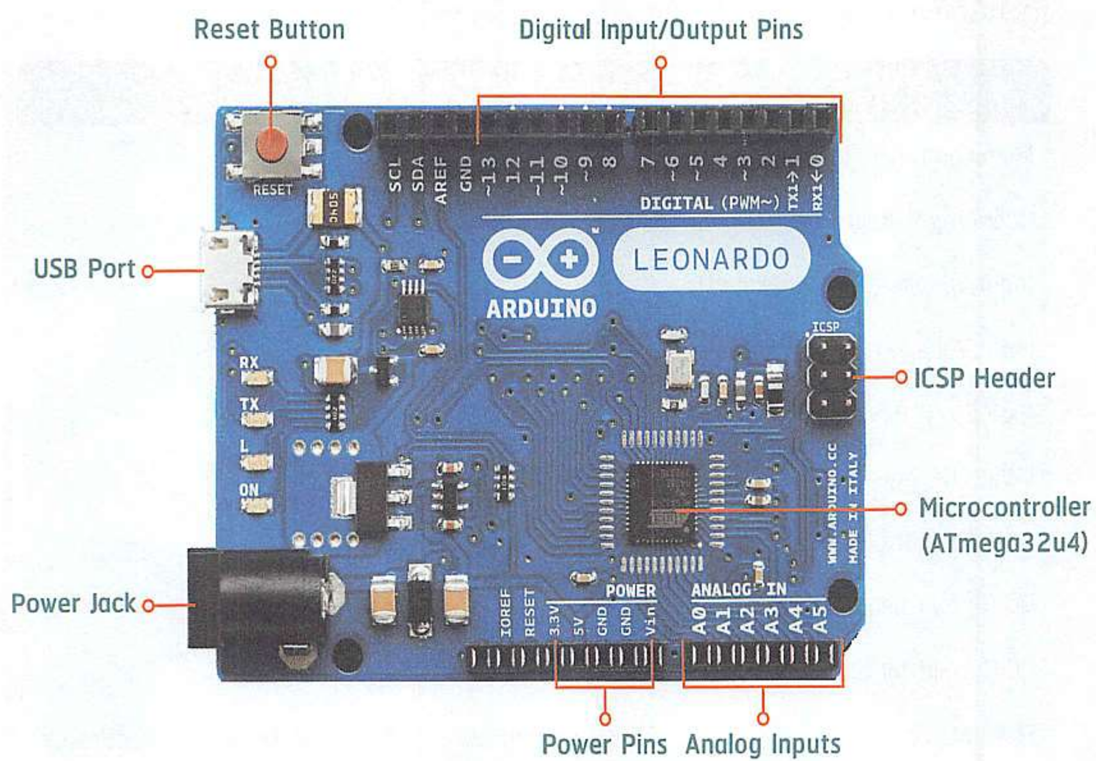


คลิป : "Which Arduino to Buy for Making Video Game Controllers?"

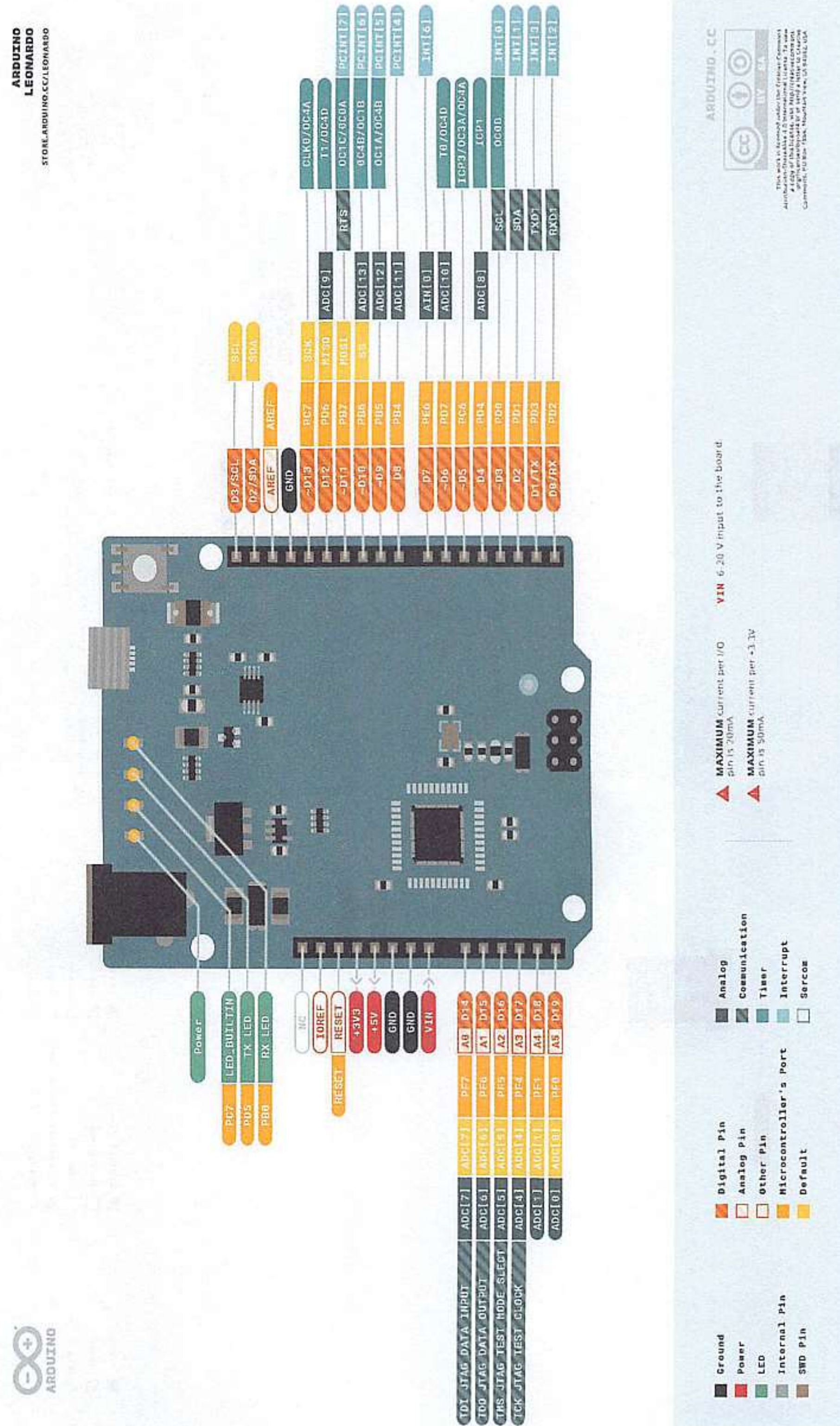
https://www.youtube.com/watch?v=kqdTL_ICt8E

Arduino LEONARDO เป็นบอร์ดที่ใช้ชิปไอซีไมโครคอนโทรลเลอร์ (Microcontroller IC Chip) รุ่น ATMEGA32U4 (Datasheet), บอร์ดประกอบด้วยขารับและขาส่งสัญญาณดิจิทัล 20 ขา (7 ขาสำหรับเอาต์พุต PWM และขารับสัญญาณอะนาล็อก 12 ขา), คริสตัลผลิตสัญญาณ 16 MHz, หัวต่อแบบ USB Jack Type Micro-B, เพาเวอร์แจ็ค, ICSP Header และปุ่ม Reset เรียกว่ามีทุกสิ่งทุกอย่างที่จำเป็นเพื่อใช้ซัพพอร์ตการทำงานของไมโครคอนโทรลเลอร์ เมื่อต่อบอร์ดกับคอมพิวเตอร์ด้วยสาย USB ก็พร้อมทำงานได้ทันที (จะต่อกับอะแด็ปเตอร์หรือใช้แบตเตอรี่แทนก็ได้)

ตารางแสดง TECH SPECS	
Microcontroller (MCU)	ATmega32u4
Operating Voltage	5 V
Input Voltage (Recommended)	7-12 V
Input Voltage (Limits)	6-20 V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.3 mm
Weight	20 g

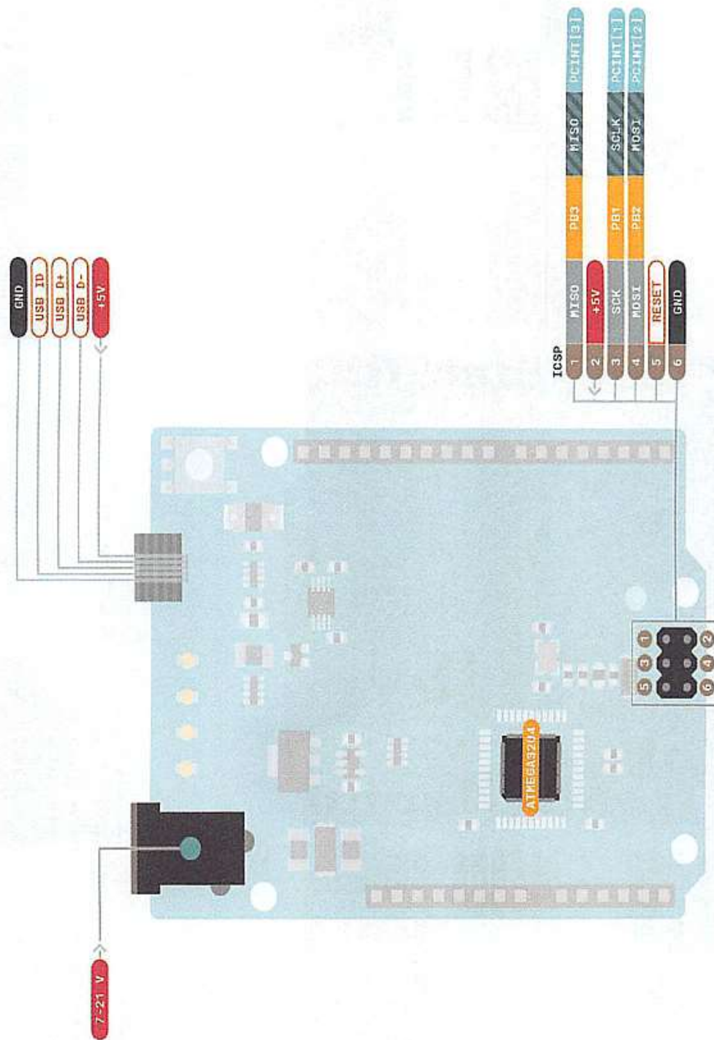


▲ รูปแสดงส่วนประกอบของบอร์ด Arduino LEONARDO



▲ รูปแสดงส่วนประกอบของบอร์ด Arduino LEONARDO Pinout - Diagram




ARDUINO
LEONARDO
STORE.ARDUINO.CC/LEONARDO
GND
USB ID
USB D+
USB D-
+5V
7-21 V
ATMEGA32U4
ICSP
1 MISO PD3 MISO PCINT[3]
2 +5V
3 SCK PB1 SCLK PCINT[1]
4 MOSI PB2 MOSI PCINT[2]
5 RESET
6 GND
Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom
MAXIMUM current per I/O pin is 20mA
MAXIMUM current per +3.3V pin is 50mA
VIN 6-20 V input to the board.
ARDUINO.CC



บอร์ด Arduino MEGA

บอร์ด Arduino MEGA 2560 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ชิป ATmega2560, มีขา Digital I/O รวม 54 ขา (15 ขาสำหรับเอาต์พุต PWM), Analog Input 16 ขา, Serial Port ชนิด UART 4 ช่อง, คริสตัลผลิตสัญญาณ (Crystal Oscillator/Clock Speed) 16 MHz, พอร์ต USB, แจ็คไฟ, หัว ICSP และปุ่ม Reset

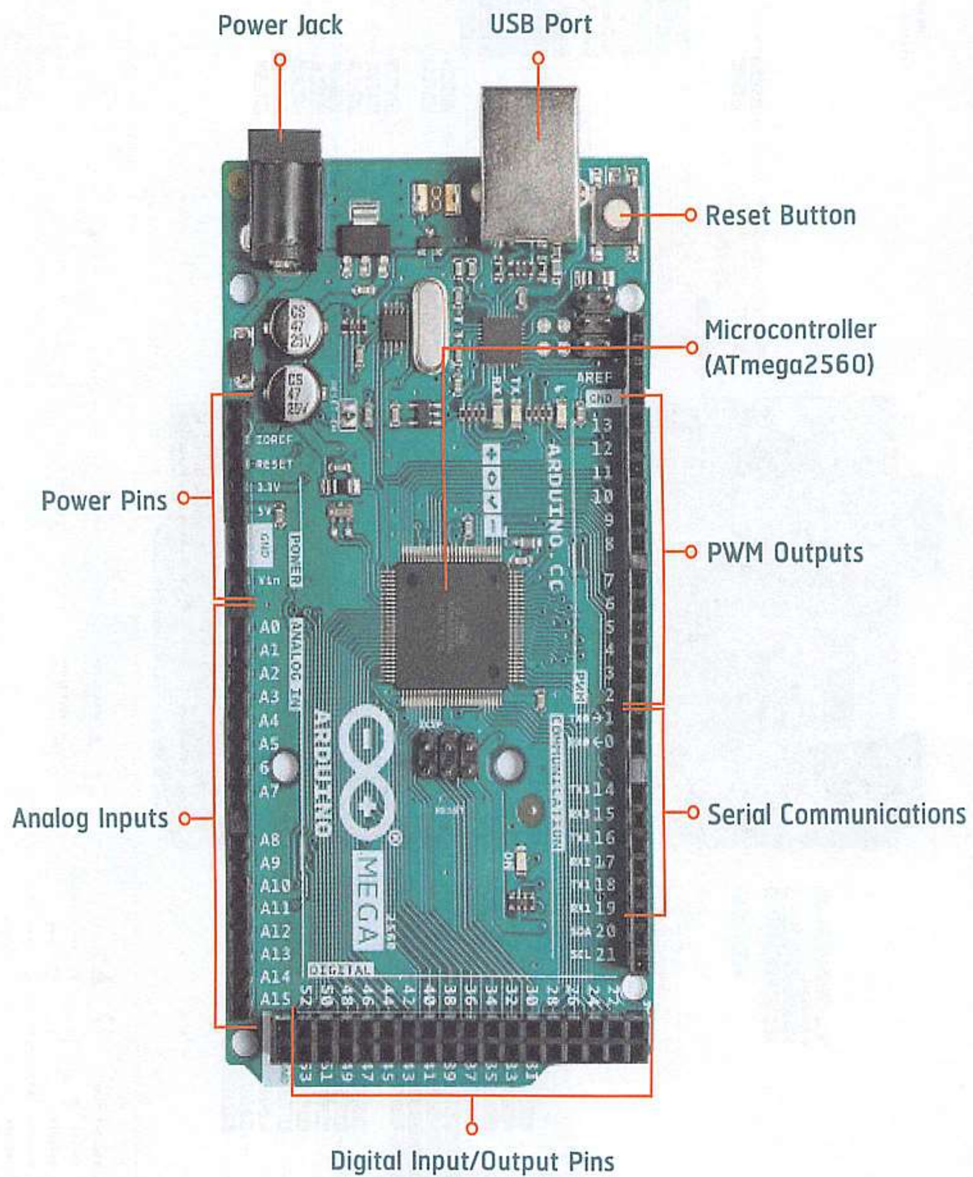
สังเกตว่าบอร์ด MEGA เป็นบอร์ดที่เพิ่มขา Pins มากขึ้น สามารถรองรับการเชื่อมต่อกับอุปกรณ์ I/O Sensors ต่างๆ ได้มากกว่าบอร์ดตระกูล UNO รวมถึงมีพื้นที่หน่วยความจำที่มากกว่าเพื่อรองรับการเขียนโปรแกรม Sketch เข้าไปได้มากกว่านั่นเอง ด้วยสเปคขนาดนี้จึงเหมาะที่จะนำไปใช้ในโปรเจ็กต์ที่มีความซับซ้อน ใครที่คิดจะทำโปรเจ็กต์ 3D Printer และ Robotic แนะนำให้เลือกใช้บอร์ดตระกูล MEGA

บอร์ด MEGA 2560 เข้ากันได้ดีกับ Shield ส่วนใหญ่ที่ออกแบบมาสำหรับ UNO และบอร์ด Duemilanove หรือ Diecimila โดย MEGA 2560 REV3 ถูกปรับปรุงใหม่เพื่อมาแทนที่บอร์ด MEGA ตัวเดิม

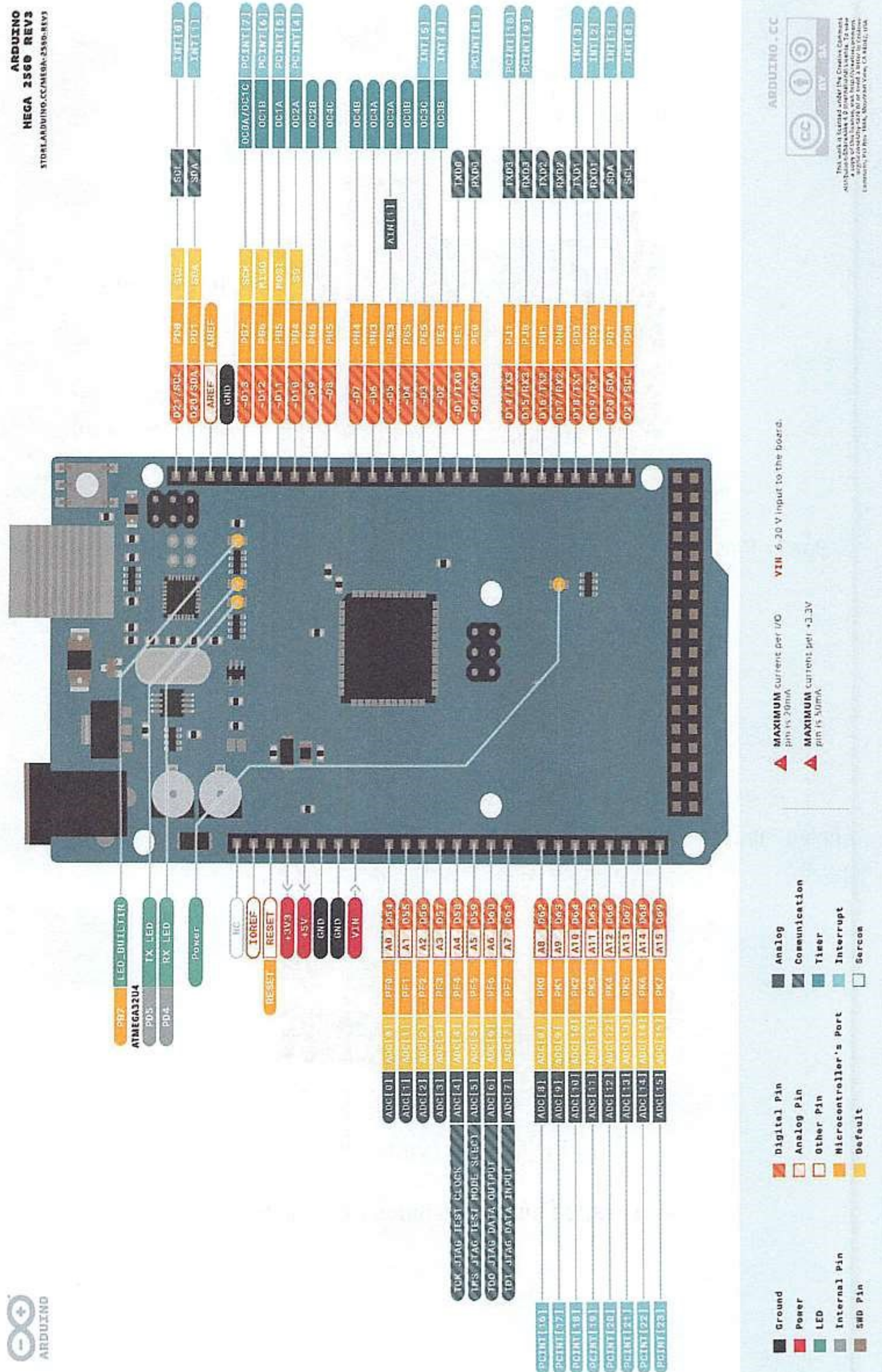


▲ รูปแสดงบอร์ด Arduino MEGA 2560 REV3 ในมุมมองต่างๆ

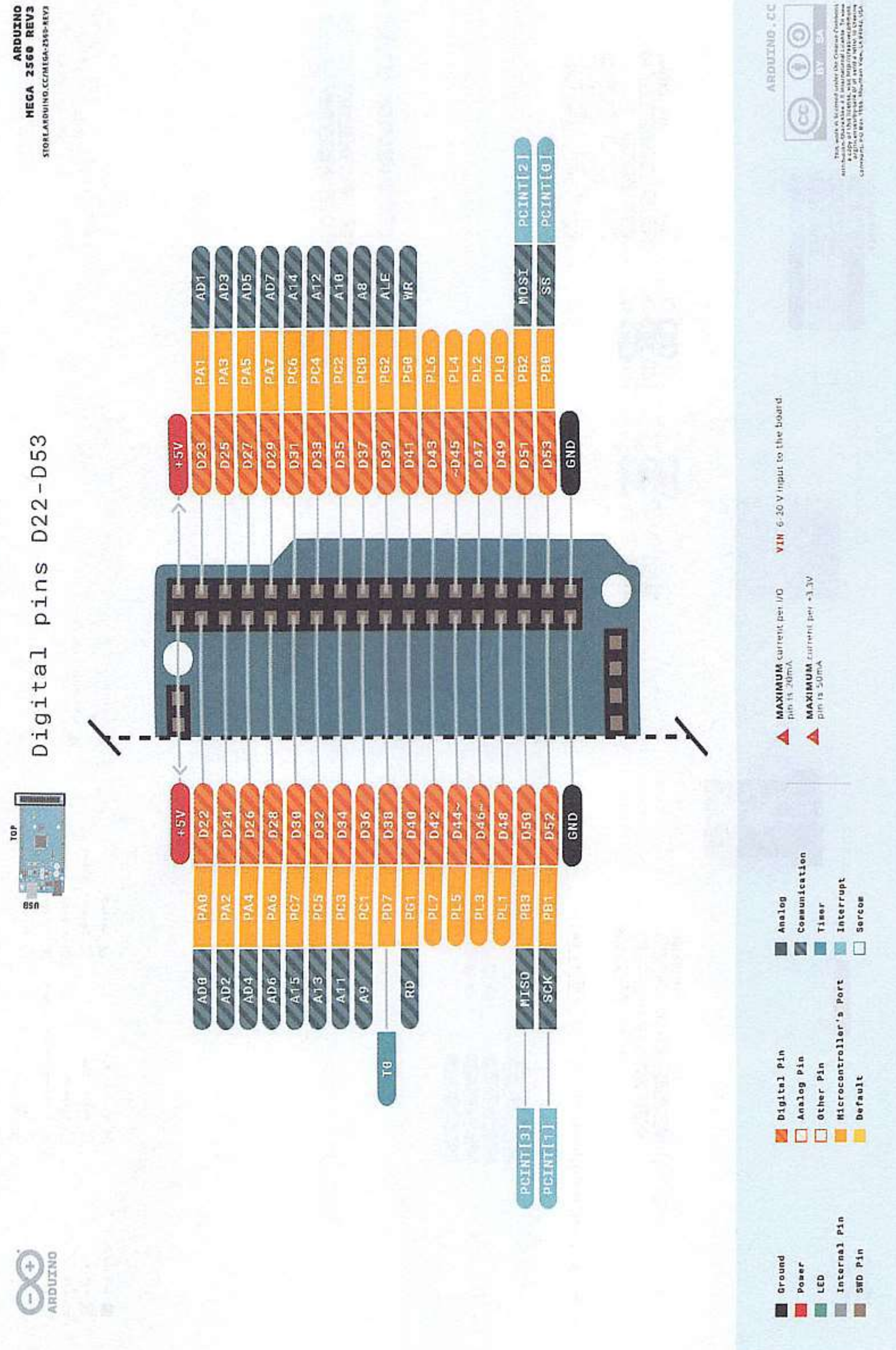
ตารางแสดง TECH SPECS	
Microcontroller (MCU)	ATmega2560
Operating Voltage	5 V
Input Voltage (Recommended)	7-12 V
Input Voltage (Limit)	6-20 V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

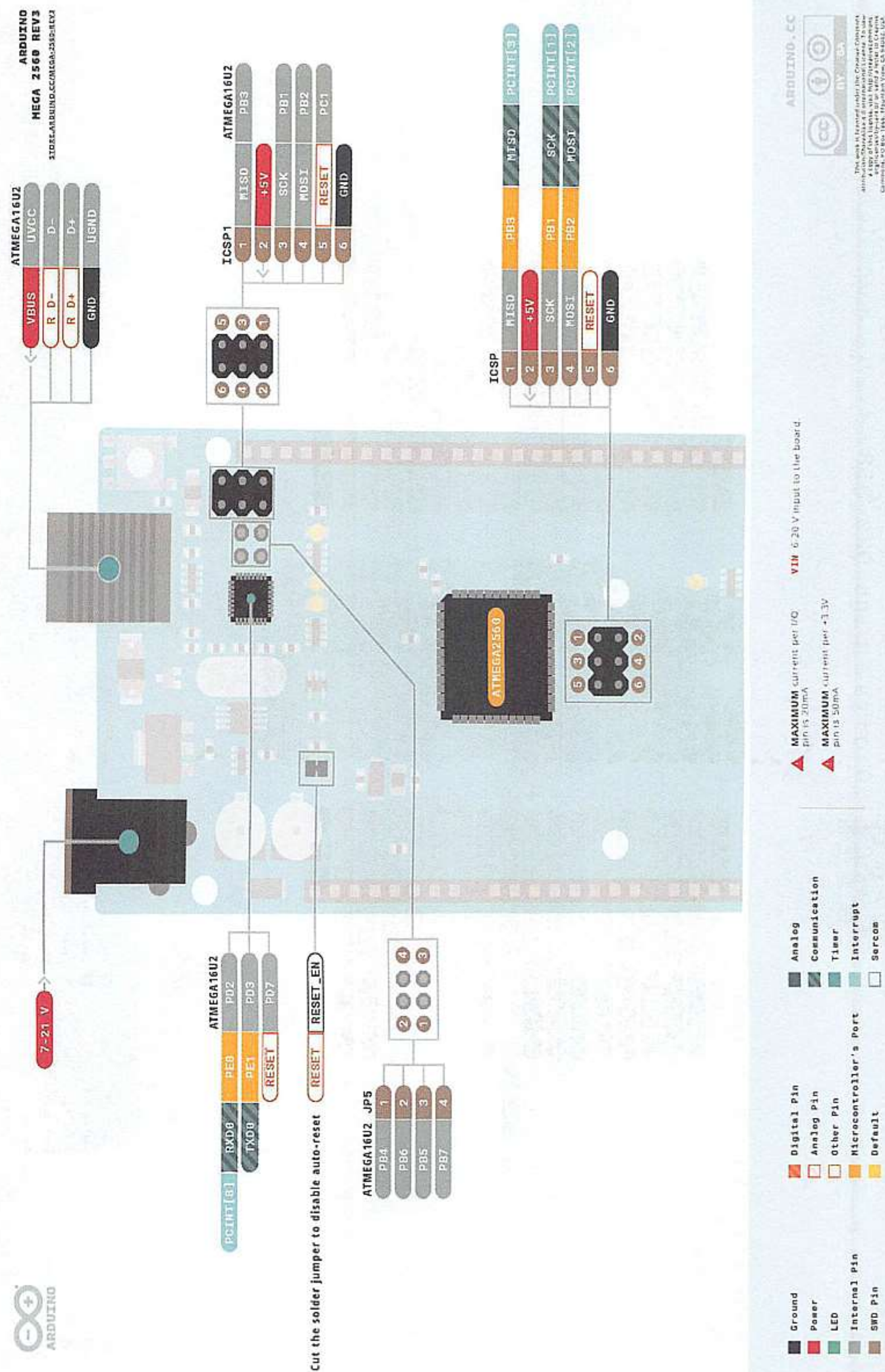


▲ รูปแสดงส่วนประกอบของบอร์ด Arduino MEGA



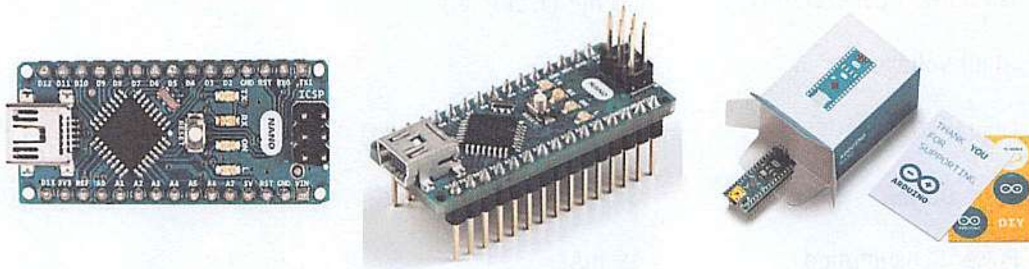
▲ รูปแสดงส่วนประกอบของบอร์ด Arduino MEGA Pinout - Diagram


ARDUINO
MEGA 2560 REV3
STORE.ARDUINO.CC/MEGA-2560-REV3
Digital pins D22-D53
TOP
USB
+5V
D22
D24
D26
D28
D30
D32
D34
D36
D38
D40
D42
D44~
D46~
D48
D50
D52
GND
D23
D25
D27
D29
D31
D33
D35
D37
D39
D41
D43
~D45
D47
D49
D51
D53
PA0
PA2
PA4
PA6
PC7
PC5
PC3
PC1
PD7
PG1
PL7
PL5
PL3
PL1
PB3
PB1
PA1
PA3
PA5
PA7
PC6
PC4
PC2
PC0
PG2
PG0
PL6
PL4
PL2
PL0
PB2
PB0
AD0
AD2
AD4
AD6
A15
A13
A11
A9
RD
MISO
SCK
AD1
AD3
AD5
AD7
A14
A12
A10
A8
ALE
WR
MOSI
SS
T0
PCINT[3]
PCINT[1]
PCINT[2]
PCINT[0]
Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom
MAXIMUM current per I/O pin is 20mA
MAXIMUM current per +3.3V pin is 50mA
VIN 6-20 V input to the board.
ARDUINO.CC

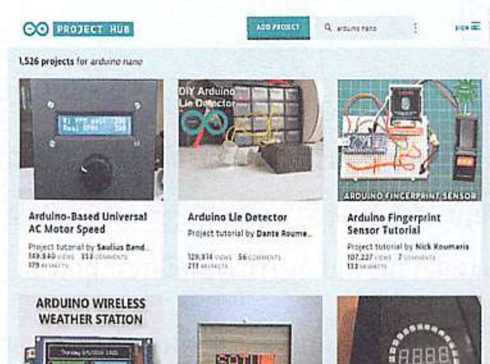

ARDUINO
ARDUINO
MEGA 2560 REV3
STORE.ARDUINO.CC/MEGA-2560-REV3
7-21 V
ATMEGA16U2
VBUS
UVCC
R D-
D-
R D+
D+
GND
UGND
ATMEGA16U2
PCINT[8]
RXD0
PE0
PD2
TXD0
PE1
PD3
RESET
PD7
Cut the solder jumper to disable auto-reset
RESET
RESET_EN
ATMEGA16U2 JP5
PB4
PB6
PB5
PB7
ICSP1
ATMEGA16U2
MISO
PB3
+5V
SCK
PB1
MOSI
PB2
RESET
PC1
GND
ATMEGA2560
ICSP
MISO
PB3
MISO
PCINT[3]
+5V
SCK
PB1
SCK
PCINT[1]
MOSI
PB2
MOSI
PCINT[2]
RESET
GND
Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom
MAXIMUM current per I/O pin is 20mA
MAXIMUM current per +3.3V pin is 50mA
VIN 6-20 V input to the board.
ARDUINO.CC

บอร์ด Arduino NANO

Arduino NANO เป็นบอร์ดขนาดจิ๋ว บอร์ดมีขนาดเล็กจนสามารถติดตั้งลงบน Breadboard ได้เลย แต่ก็มีคุณสมบัติพร้อมสำหรับประดิษฐ์อุปกรณ์ DIY ได้มากมาย เพราะบอร์ด NANO ใช้ชิปไมโครคอนโทรลเลอร์รุ่น ATmega328 (Arduino NANO 3.x) ซึ่งเป็นรุ่นเดียวกับบอร์ด UNO ฟังก์ชันการทำงานจึงไม่แตกต่างกัน ถ้าเป็นรุ่นที่ใช้ชิปรุ่น ATmega328P สังเกตว่าจะมีตัว ‘P’ ต่อท้าย แสดงว่าเป็นรุ่นที่กินไฟน้อย (Low-Power Microcontroller) โดยอาศัยเทคโนโลยี picoPower คือ มีโหมดประหยัดพลังงาน บอร์ด NANO จะตัด DC Jack ที่เปลืองพื้นที่ออกไป บอร์ดจะรับไฟผ่านช่อง Mini-B USB ซึ่งมีขนาดเล็กกว่าช่องมาตรฐาน แล้วก็มีการตัดอีกหลายๆ ส่วนออกไป เพื่อให้บอร์ดมีขนาดเล็กที่สุดเท่าที่จะทำได้ เหมาะจะใช้ในโปรเจ็กต์ขนาดเล็ก ราคาบอร์ดแท้ Arduino NANO (Board from Italy) อาจจะมีราคาสูง แต่สามารถหาบอร์ดทดแทน (Arduino Compatible Board) มาใช้งานแทนได้



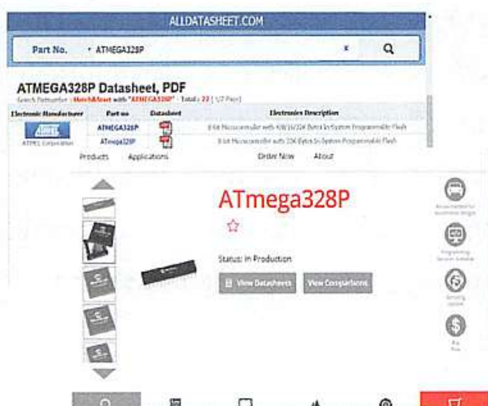
▲ รูปแสดงบอร์ด Arduino NANO ในมุมมองต่างๆ



▲ รูปแสดงการค้นหา Arduino NANO Projects ใน Arduino Project Hub

สำหรับตัวอย่างการนำ Arduino NANO ไปใช้งาน แนะนำให้เข้าไปค้นหาที่ Arduino Project Hub ซึ่งจะอยู่ในส่วนของ https://create.arduino.cc แล้วใช้คำค้นหา “arduino nano” ก็จะพบโปรเจ็กต์มากมาย สังเกตว่ามีจำนวนมากถึง 1,526 Projects (ขณะทำหนังสือ) แนะนำให้คลิกหัวข้อโปรเจ็กต์ต่างๆ ดู จะได้รู้ว่า Arduino NANO สามารถนำไปพัฒนาเป็นอุปกรณ์ DIY เพื่อใช้งานอะไรได้บ้าง

ตารางแสดง TECH SPECS	
Microcontroller (MCU)	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

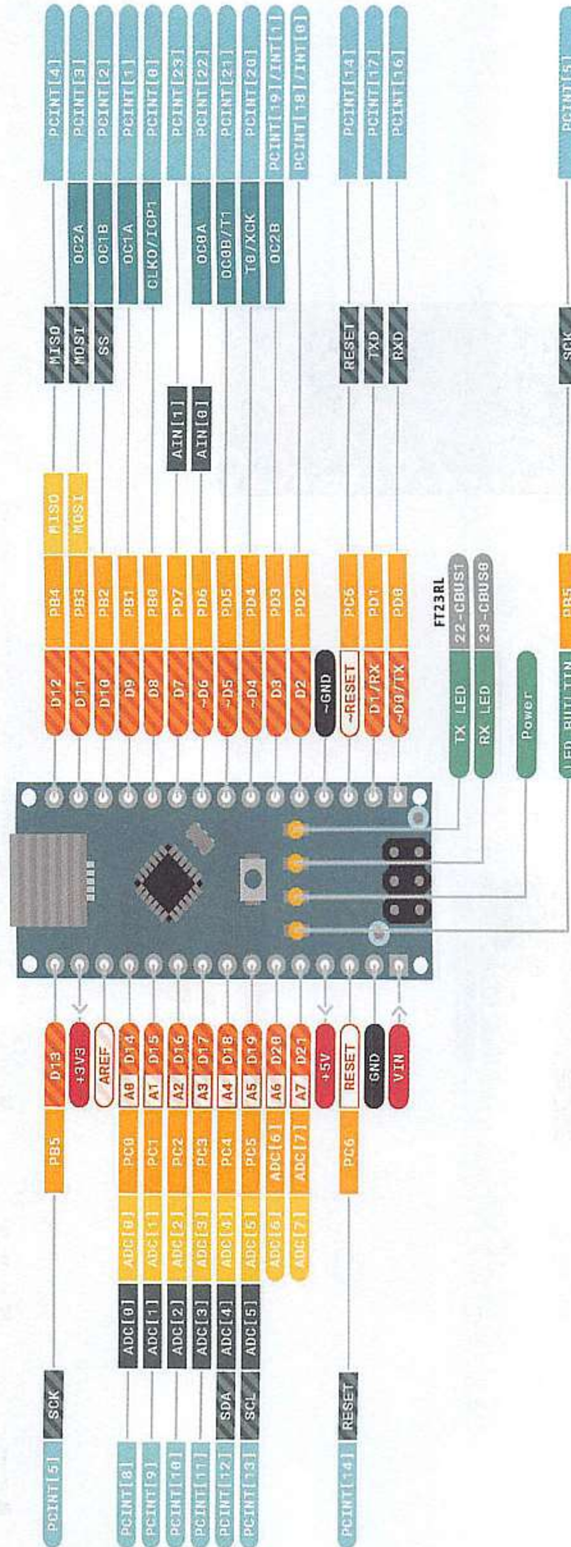


ถ้าอยากเรียนรู้เพิ่มเติมเกี่ยวกับ MCU แนะนำให้ค้นหาได้ที่เว็บไซต์ www.alldatasheet.com ซึ่งเป็นแหล่งรวบรวมเอกสารทางเทคนิคจากทุกค่าย

หรือเข้าไปที่เว็บไซต์ของผู้ผลิตชิปโดยตรง ซึ่งในที่นี้คือ www.microchip.com

◀ รูปแสดงการค้นหา Datasheet

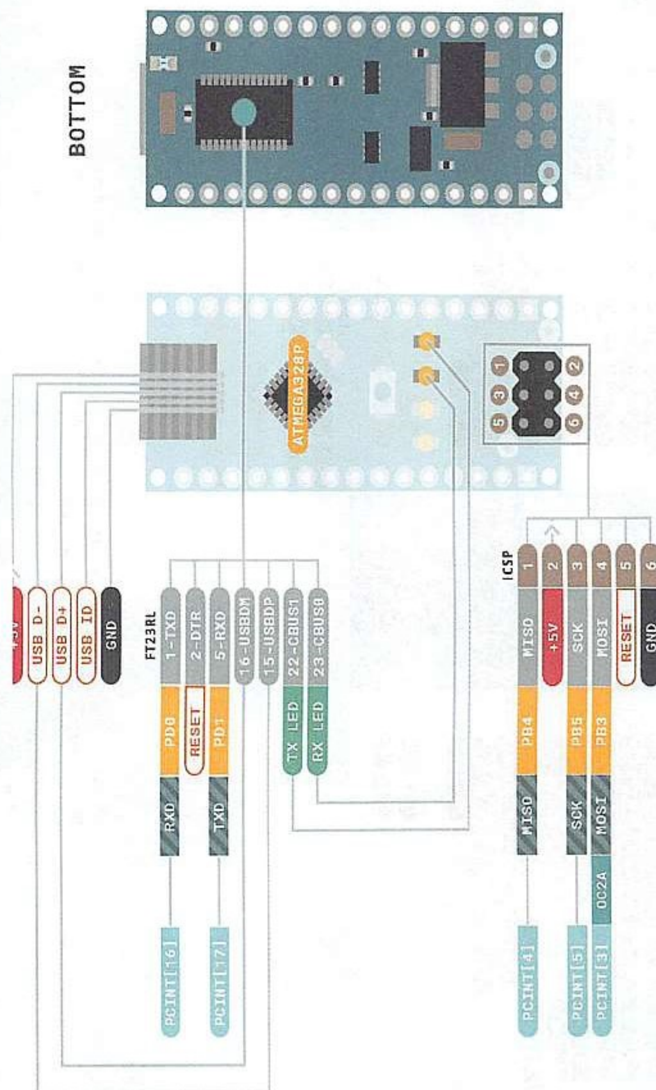






รูปแสดงส่วนประกอบของบอร์ด Arduino NANO Pinout - Diagram

ARDUINO NANO
STORE.ARDUINO.CC/NANO



ARDUINO

Ground
Power
LED
Internal Pin
SWD Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

Analog
Communication
Timer
Interrupt
Sercom

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board

ARDUINO.CC



ข้อมูล Sensors/Modules, Motors และ Shields

ประเภทของ Sensors/Modules

Arduino Sensors/Modules เป็นอุปกรณ์ต่อพ่วงเพื่อรับคำสั่งจากบอร์ด Arduino ให้ทำงานตามต้องการ โดย Sensors/Modules ที่ใช้ได้กับ Arduino แบ่งตามลักษณะการใช้งานดังนี้

Sensors/Modules	Examples			
เซนเซอร์ความเร่ง/ไจโร/IMU (Accelerometer, Gyroscope & IMU Sensors)	Accelerometer (MMA7361)	Gyroscope (GY-61 ADXL335)	IMU (GY-9255 IMU 9DOF)	Digital Compass Magnetometer (MAG3110)
เซนเซอร์วัดระยะทาง (Distance/Range Sensors)	Ultrasonic Sensor (HY-SRF05)	Infrared Distance Sensor (GP2Y0A710K0F)	Ambient Light Sensor (VL6180)	Laser Ranging Distance Sensor (VL53L0X V2)
เซนเซอร์แสงและการมองเห็น (Optical & Vision Sensors)	Camera Module (OV7670)	Infrared Barrier Module (OV7670)	LDR Photoresistor Sensor Module (OV7670)	Laser Module (OV7670)
เซนเซอร์ตรวจจับการเคลื่อนไหว (Motion Sensors)	Vibration Sensor (801S)	Microwave Motion Sensor (HB100)	PIR Motion Sensor (HC-SR501)	Knock Sensor (KY-031)

Sensors/Modules	Examples			
เซนเซอร์วัดสภาพแวดล้อม (Environmental Sensors)	Temperature and Humidity Sensor (DHT111)	Color Recognition Sensor (GY-33))	Flame Sensor (5-Channel)	Soil Moisture Sensor (Red PCB Keysl)
เซนเซอร์แก๊ส (Gas Sensors)	Gas Detection Sensor (MQ-135)	CO2 NDIR Gas Sensor (MH-Z14)	Alcohol Sensor (MQ-3)	CO Gas Sensor (MQ-7)
โมดูลวัดแรงดันและกระแสไฟฟ้า (Voltage & Current Sensors)	Digital DC Voltmeter (0.28" 0-100 V)	Switching Power Supply Module (AC-DC step-down 5V 700 mA)	Current Sensor Module (ACS712-30A)	Battery Power Indicator (11.1-12.6 V)
โมดูลสวิตช์ (Switch Modules)	Switch Digital Touch Capacitive Module (4-way)	Keypad Module (4x4 Matrix)	Rocker Switch (Catalex)	Capacitive Touch Switch (Catalex)
โมดูลบันทึกข้อมูล (Record Modules)	SD Card Module	Data Logger Shield	I2C Interface EEPROM Module (AT24C256)	SPI FLASH Storage Module (W25Q32)
โมดูลนาฬิกา Real Time Clock (Clock Modules)	Real Time Clock Module (DS1307)	Tiny RTC I2C Modules (DS1307)	Precision Clock Module (DS3231)	High Precision Clock Module (DS1307)

Sensors/Modules	Examples			
โมดูลสื่อสารไร้สาย (Wireless Modules)	Wireless Module (NRF24L01)	Bluetooth Module (CC2541)	iBeacon Bluetooth Module (CC2541)	FM Radio Transmitter Module
โมดูลอินเทอร์เน็ต (Internet Modules)	SPI Interface Ethernet Network Module (ENC28J60)	Ethernet Shield (ENC28J60)	Ethernet Shield (W5100 R3)	Ethernet Module (W5100)
โมดูล RFID/NFC/Smart Card (RFID/NFC/ Smart Card Modules)	NFC/RFID Module (PN532)	RFID Tag	RFID Reader Module (RDM6300)	RFID USB Reader
โมดูล USB Converter (USB Converter Modules)	USB to Serial RS-232 Converter	USB to Serial TTL (CP2102)	RS232 to TTL (MAX3232)	USB 2.0 To TTL UART Module (CP2104)
โมดูล JoyStick (JoyStick Modules)	Arcade Analog Joystick	JoyStick Shield	PS2 XY Joystick Module	Rotary Encoder Module
โมดูลเสียง (Sound Sensors)	Active Buzzer Module	High Sensitive Sound Microphone Module	MP3 Module (VS1003)	Voice Board Module (ISD1820)

Sensors/Modules	Examples			
โมดูลเครื่องเสียง (Amplifier Modules)	Power Amplifier with Volume Stereo (PAM8406)	Amplifier Class D (TDA8932)	Audio Power Amplifier (VS88715V)	Amplifier Board (TPA3116D2)
โมดูล GSM/GPS (GSM/GPS Modules)	Voice Decoding Module Phone (DTMF MT8870)	GSM/GPRS&GPS/ BDS Development Board (A9G)	Ublox NEO-M8N GPS Module with Antenna	SIM900 GSM/ GPRS GA6 Module
โมดูล 3D Printer/CNC (3D Printer & CNC Modules)	3D Printer Endstop Switch (RAMPS 1.4)	3D Printer Smart Controller with LCD (RAMPS 1.4)	CNC Shield V4 Kit	Stepper Motor Drive (DRV8825)
โมดูลกำเนิดสัญญาณ (Signal Generator Modules)	Signal Generator (ICL8038)	Frequency Adjustable Pulse Generator Module (N555)	Square Wave Signal Generator Frequency & Duty Cycle Adjustable (N555)	Arduino Compatible Signal Generator (Tsunami)
โมดูลแปลงสัญญาณ I/O (I/O Converter Modules)	Optocoupler Module PLC Signal Level Voltage Converter	SPI CAN Bus Controller and Driver Module	Frequency to Voltage Converter	A/D Module

Credit : www.trossenrobotics.com และ www.arduinoall.com

ประเภทของ Arduino Motors และ Motor Driver Modules

Motors/Motor Driver Modules เป็นอุปกรณ์มอเตอร์และวงจรขับมอเตอร์เพื่อใช้ควบคุมการเคลื่อนที่ สามารถนำมาใช้ได้กับบอร์ด Arduino แบ่งตามลักษณะการใช้งานดังนี้

Motors/Driver Module	Examples			
Servo Motor	Servo Motor (SG90)	Servo Motor (S3003)	Servo Motor (ES08MA)	Servo Motor (MG996R)
Stepper Motor	Stepper Motor 5 V 4-Phases (28BYJ-48)	Stepper Motor 12 V 4-Phases (28BYJ-48)	Stepper Motor 12 V (17HS3401)	Stepper Motor 12 V (17HS8401)
DC Motor	DC Motor 3-6 V	Mini DC Motor 3-6 V	Micro Vacuum Pump 370 DC 3V	DC Vibration Motor
Motor Driver Modules	16-Channel 12-bit PWM Servo Shield I2C Interface (PCA9685)	USB to Serial TTL (CP2102)	Dual DC Motor Driver Module (TB6612FNG)	4 DC Motor Driver Module (L293D)

ประเภทของ Arduino Shields

Arduino Shields เป็นอุปกรณ์เสริมใช้สำหรับต่อพ่วงกับบอร์ด Arduino โดยการสวมเข้ากับ Digital Pin และ Analog Pin ซึ่งชิลด์เหล่านี้ส่วนใหญ่มาพร้อมกับ Library ที่ช่วยให้สามารถใช้งานชิลด์ได้ง่าย ไม่ต้องยุ่งยากกับการเขียนคำสั่งควบคุมเอง แบ่งตามลักษณะการใช้งานดังนี้

Arduino Shields	Examples	
WIFI/ETHERNET	Arduino WiFi Shield 101	Ethernet Shield for Arduino (W5500)
BLUETOOTH	Adafruit Bluefruit LE Shield - Bluetooth LE for Arduino	1Sheeld+ for iOS and Android
DATA LOGGING	Adafruit Assembled Data Logging Shield	Adafruit Ultimate GPS Logger Shield

Arduino Shields	Examples	
TFTS & DISPLAYS	LCD Shield Kit w/16x2 Character Display - BLUE AND WHITE	2.8" TFT Touch Shield for Arduino with Resistive Touch Screen
GPS	Adafruit Ultimate GPS Logger Shield - Includes GPS Module	Adafruit GPS Logger Shield Kit - v1.1
MOTOR/STEPPER/ SERVO	Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit - v2.3	Adafruit 16-Channel 12-bit PWM/Servo Shield - I2C Interface
GAMES	DIY Gamer Kit from Technology Will Save Us	Gameduino Shield

Arduino Shields	Examples	
LEDS	Adafruit NeoPixel Shield for Arduino - 40 RGB LED Pixel Matrix	Adafruit RGB Matrix Shield for Arduino
PROTOTYPING	Adafruit Proto Shield for Arduino Unassembled Kit - Stackable - Version R3	Adafruit Proto Shield for Arduino Kit - v.5
SOUND/MUSIC	Adafruit "Music Maker" MP3 Shield for Arduino (MP3/Ogg/WAV)	Music & Sound Add-on Pack for Arduino - v1.1
POWER	Adafruit PowerBoost 500 Shield - Rechargeable 5 V Power Shield	

Arduino Shields	Examples
NFC/RFID	Adafruit PN532 NFC/RFID Controller Shield for Arduino + Extras
TOUCH	Adafruit 12 x Capacitive Touch Shield for Arduino - MPR121

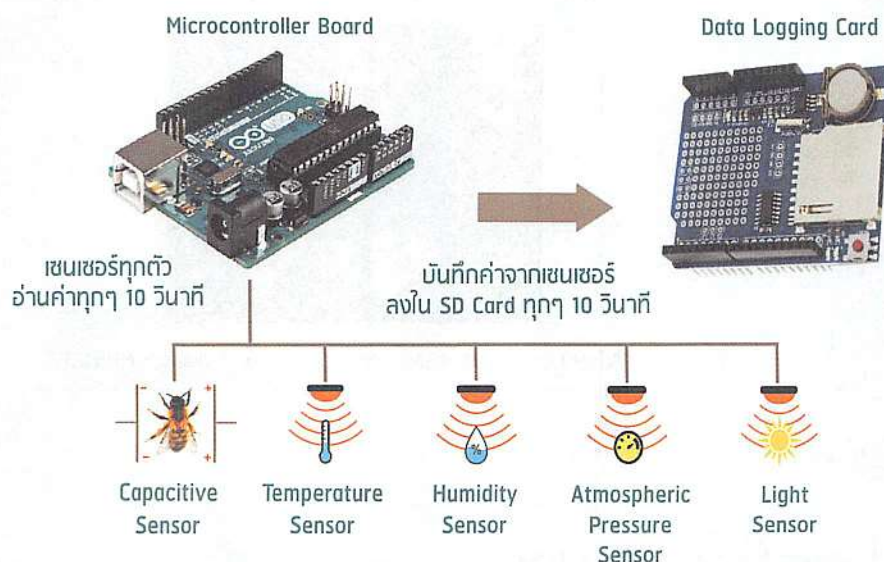
Credit : www.adafruit.com

ตัวอย่างการใช้งานบอร์ด Arduino กับ Sensors

ในการใช้งานบอร์ด Arduino โดยส่วนใหญ่จะใช้งานร่วมกับอุปกรณ์ Sensors แบบต่างๆ เพื่อควบคุมและสั่งการให้ทำงานตามที่ต้องการ ขออธิบายผ่านโปรเจ็กต์ที่ชื่อว่า "Monitoring Solitary Bees Using Open Technology" ซึ่งเป็นโครงการทางวิทยาศาสตร์ในการมอนิเตอร์พฤติกรรมการทำรังของผึ้งเมสันในสวนหลังบ้าน เป็นโปรเจ็กต์ของนาย Mike Teachman มือสมัครเล่นที่ชื่นชอบผึ้ง กับนาย Paul Perrault วิศวกรอาวุโสทางแอปพลิเคชัน

ผึ้งเมสัน (Mason Bee) จะมีพฤติกรรมที่แตกต่างจากผึ้งน้ำหวาน หรือฮันนี่บี (Honey Bee) โดยเมสันจะเป็นผึ้งที่ชอบอยู่ตามลำพัง มันไม่ใช่สัตว์สังคม ผึ้งตัวเมียจะมีความสมบูรณ์ ไม่มีผึ้งงาน พวกมันใช้ส่วนท้องขนเกสร พวกมันชอบทำรังในรูซึ่งมีส่วนสำคัญต่อการผสมเกสรของไม้ผลนานาชนิด

เบื้องหลังไอเดียของโปรเจ็กต์นี้คือ การเฝ้าติดตามดูพฤติกรรมของผึ้งเมสันที่แต่ละตัวจะอาศัยอยู่ตามลำพังแบบรังละตัว โดยใช้เซนเซอร์ติดไว้กับรังผึ้งทุกรัง ไม่ได้ติดเซนเซอร์เข้ากับตัวผึ้งแต่อย่างใด โดยเซนเซอร์ที่ใช้จะเป็นแบบ Capacitive Sensor ที่ใช้ตรวจจับความเคลื่อนไหวของผึ้งเมสันที่บินเข้าออกจากรัง และใช้บอร์ดไมโครคอนโทรลเลอร์ Arduino UNO เป็นระบบควบคุมการทำงาน ต่อมาได้มีการขยายขอบเขตการศึกษาเพิ่มเติม จากในช่วงแรกที่เราตรวจจับแค่การบินเข้าบินออกจากรังเท่านั้น แต่ได้เพิ่มการวัดกิจกรรมของผึ้งที่แสดงความรู้สึกออกมาได้อีกด้วย แต่จะต้องใช้เซนเซอร์ชนิดอื่นๆ เพิ่มเติม พร้อมกับการพัฒนาระบบการวัดที่จะใช้ในการรวบรวมข้อมูลจำนวนมาก



▲ รูปแสดงแผนภาพการทำงานของเครื่องมอนิเตอร์พฤติกรรมของผึ้งเมสัน

จากรูป เราสามารถแบ่งการทำงานออกเป็น 3 ส่วน ดังนี้

1. ส่วน Input (Sensors) ใช้สำหรับรับข้อมูลที่ตรวจวัดได้จากเซนเซอร์ ซึ่งมี 5 ชุดข้อมูล ได้แก่
    - Capacitive Sensor ใช้ตรวจจับวัตถุ ในที่นี้คือ ผึ้ง
    - Temperature Sensor ใช้ตรวจจับอุณหภูมิ
    - Humidity Sensor ใช้ตรวจจับความชื้น
    - Atmospheric Pressure Sensor ใช้วัดความดันอากาศ
    - Light Sensor ใช้วัดความเข้มแสง

2. **ส่วน Process (Arduino Board)** ใช้สำหรับประมวลผลและสั่งการให้อุปกรณ์ต่างๆ ทำงานตามที่ได้เขียนโปรแกรมผ่านทาง Arduino IDE โดยไมโครคอนโทรลเลอร์จะรับคำสั่งให้นำข้อมูล Input ของทั้ง 5 Sensors มาประมวลผล ได้แก่ ข้อมูลตรวจจับวัตถุ อุณหภูมิ ความชื้น ความดันอากาศ และความเข้มแสง มาบันทึกและเก็บใส่ในส่วน Output ในทุกๆ 10 วินาที

3. **ส่วน Output (Data Logging Card or Output Hardware)** ใช้สำหรับเก็บข้อมูลหรือแสดงผลที่ได้จากเขียนโปรแกรมผ่านการประมวลผลของบอร์ด Arduino ในที่นี้ Data Logging Card (SD Card) จะบันทึกข้อมูลทั้งหมดในทุกๆ 10 วินาที



▲ รูปแสดงตัวเครื่องมอนิเตอร์พฤติกรรมผึ้งที่ประดิษฐ์ขึ้นใช้งานจริง



▲ คลิปแสดงการทำงาน BeeActivityClips

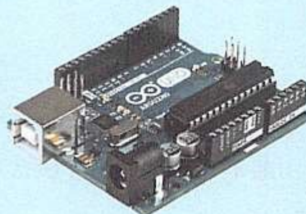
https://www.youtube.com/watch?v=R4sbuPAROJc&feature=emb_title

Credit : https://blog.arduino.cc

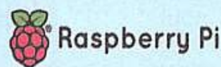


Arduino vs Raspberry Pi บอร์ดไหนดีกว่ากัน?





VS





หลายคนอาจจะเคยได้ยินบอร์ดที่ชื่อ "Raspberry Pi" ที่ได้รับความนิยมเช่นเดียวกับบอร์ด Arduino จนมีคำถามในใจว่า บอร์ดตัวไหนดีกว่ากัน หรือบอร์ดทั้งสองแตกต่างกันอย่างไร สรุปให้พอเข้าใจง่ายๆ ได้ดังนี้

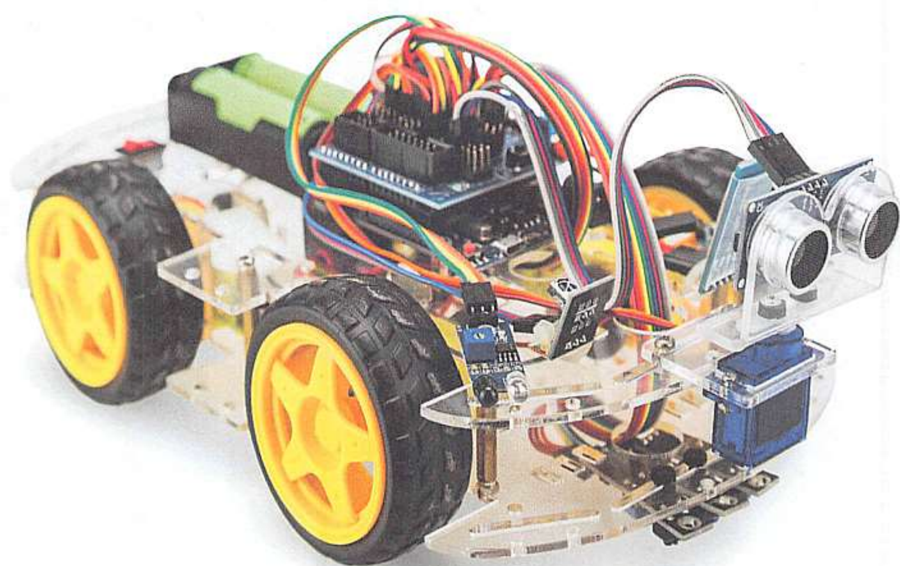
Arduino คือ บอร์ดไมโครคอนโทรลเลอร์ที่เน้นหน้าที่ในการรับส่งข้อมูลผ่านขา (Pins) ในการเชื่อมต่อกับเซนเซอร์ เน็ตเวิร์ค อินเทอร์เน็ต และอุปกรณ์ต่อพ่วงอื่นๆ ไม่เน้นงานที่ละเอียดซับซ้อน เนื่องจากมีขนาดของหน่วยความจำที่จำกัด

Raspberry Pi คือ คอมพิวเตอร์เต็มรูปแบบที่มีทั้งซีพียู เมมโมรี และเน็ตเวิร์ค ที่สามารถติดตั้งระบบปฏิบัติการ (OS) แล้วกลายเป็นเครื่องคอมพิวเตอร์ขนาดเล็กตัวหนึ่ง ส่วนใหญ่นิยมนำมาใช้ทำเครื่องคอมพิวเตอร์ราคาถูกสำหรับนักเรียน เนื่องจากมีการใส่ GPIO (General Purpose Input/Output) หรือพอร์ตอเนกประสงค์ ซึ่งเป็นตัวรับส่งสัญญาณดิจิทัล (Digital Input/Output) จึงสามารถนำไปควบคุมอุปกรณ์ภายนอกต่างๆ ได้เหมือนกับ Arduino แต่ประสิทธิภาพสูงกว่า เนื่องจากรองรับระบบปฏิบัติการเต็มรูปแบบ ทำให้รองรับการเขียนโค้ดที่มีความซับซ้อน และต้องการการประมวลผลที่สูงกว่า

สรุปว่า ถ้าใช้สำหรับการเรียนรู้เพื่อสร้างโปรเจ็กต์แบบง่ายๆ เพื่อใช้ในการควบคุมอุปกรณ์พื้นฐานต่างๆ ที่มีระดับความซับซ้อนไม่สูงมาก Arduino จะเหมาะสมกว่า เนื่องจากเป็นวงจรสำเร็จรูปที่มีอุปกรณ์ต่อพ่วงสนับสนุนการใช้งานที่หลากหลาย ใช้งานง่าย และราคาถูกกว่า เว้นแต่ต้องการใช้งานบอร์ดที่มีความทัดเทียมกับคอมพิวเตอร์ เพื่อมารองรับการทำงานที่มีความซับซ้อนสูง ก็ให้เลือกใช้ Raspberry Pi ที่มีพลังในการประมวลผลที่สูงกว่า

บทสรุปท้ายบท

บทนี้เราได้ศึกษา Microcontroller Hardware และ Arduino Board Family ตลอดจนอุปกรณ์เสริมต่างๆ ซึ่งเนื้อหาทั้งหมดเป็นเพียงการหยิบยกมาเป็นตัวอย่างเท่านั้น เนื่องจากเราเน้นพื้นฐานเพื่อนำไปสู่การใช้งานมากกว่า ทฤษฎีจึงไม่เข้มข้น กล่าวถึงพอสังเขปเกี่ยวกับสถาปัตยกรรมภายในไมโครคอนโทรลเลอร์ แสดงภาพไดอะแกรมของ Arduino Board และแนะนำให้รู้จักบอร์ดเพียงบางรุ่นเท่านั้น แต่หากได้ศึกษาในบทนี้จนเข้าใจแล้ว ก็สามารถทำความเข้าใจบอร์ดรุ่นอื่นๆ ได้ไม่ยาก เพราะคุณสมบัติพื้นฐานจะคล้ายๆ กัน แต่อาจเพิ่มคุณสมบัติพิเศษเพื่อให้เหมาะกับงานพัฒนาโปรเจ็กต์เท่านั้น สำหรับอุปกรณ์เสริมก็เช่นเดียวกัน เราเพียงชี้ให้เห็นว่าบอร์ด Arduino นั้นมีอุปกรณ์ซัพพอร์ตมากมายเพื่อใช้พัฒนาชิ้นงานหรือสร้างผลิตภัณฑ์ เพราะฉะนั้น หากผู้อ่านต้องการศึกษาลงลึกในเรื่องไหน ก็สามารถค้นคว้าเพิ่มเติมได้ด้วยตนเอง ทั้งนี้ผู้เขียนได้แนะนำคำค้นหาและใส่ลิงค์ไว้ให้ศึกษาเพิ่มเติมไว้ด้วย

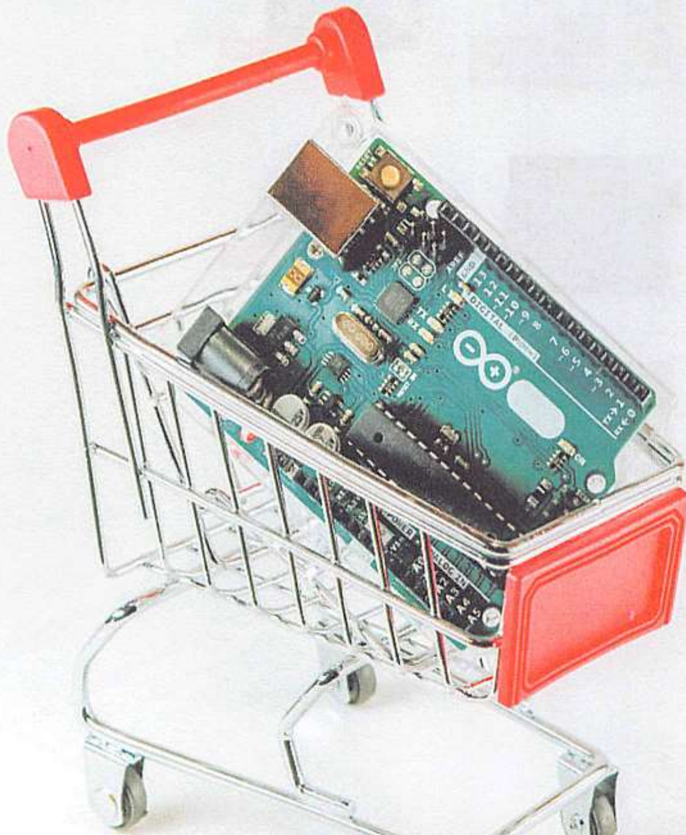


CHAPTER

03

การเลือกซื้อ Arduino ให้เหมาะกับการใช้งาน

จากอดีตจนถึงปัจจุบัน ผลิตภัณฑ์ Arduino นั้นถูกผลิตออกมาเป็นจำนวนมาก ฉะนั้น ในการเลือกซื้อ Arduino มาใช้งาน จำเป็นจะต้องศึกษา Arduino Products ชุดต่างๆ ให้ดีเสียก่อนว่า ผลิตภัณฑ์มีกี่กลุ่ม แต่ละกลุ่มเหมาะกับใคร เรียนรู้ว่ารุ่นไหนเหมาะจะใช้กับงานลักษณะไหน ไปดูตารางเปรียบเทียบสเปคเพื่อให้เห็นความแตกต่างชัดเจนมากขึ้น การตรวจสอบบอร์ดแท้ทำอย่างไร ทั้งหมดที่กล่าวมาก็เพื่อให้ผู้อ่านมีความรอบรู้อย่างเพียงพอ สามารถพิจารณาเลือกซื้อบอร์ดได้ด้วยตนเอง

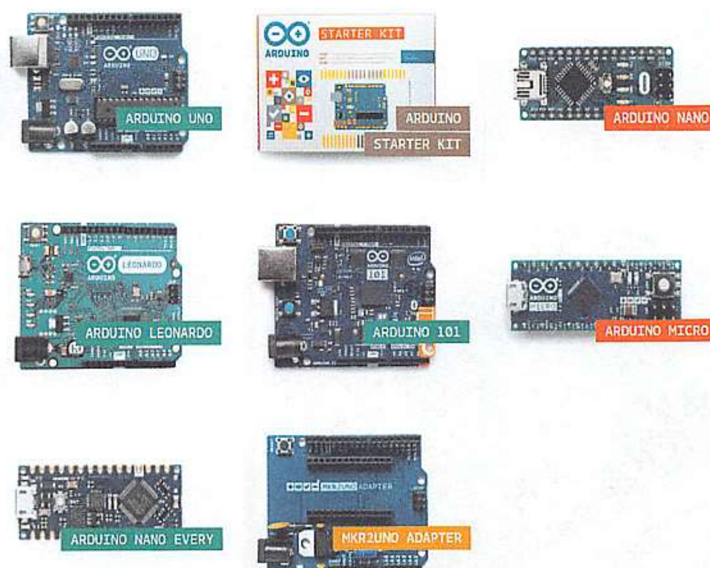


สินค้ากลุ่มไหนเหมาะกับใคร (Arduino Products)

หากต้องการดูผลิตภัณฑ์ทั้งหมดของ Arduino ที่ผลิตจำหน่ายอย่างเป็นทางการ ได้แก่ Boards, Modules, Motors & Servos, Shields, Kits และ Accessories อื่นๆ สามารถเข้าไปสำรวจดูสินค้าต่างๆ ที่กล่าวมาได้ที่เว็บไซต์ www.arduino.cc เลือกเมนู **STORE** ซึ่งเป็นหน้าร้านแสดงรายการสินค้าทั้งหมด

หรือหากต้องการศึกษารายละเอียด เช่น การแบ่งหมวดหมู่สินค้า การเปรียบเทียบสเปคของสินค้าแต่ละรุ่น หรือหากมีข้อสงสัยว่า บอร์ด Arduino ของคุณเป็นของแท้หรือเทียม ก็สามารถเรียนรู้วิธีสังเกตบอร์ดเทียมได้ในบทนี้ ซึ่งผู้เขียนได้สรุปเนื้อหามาจากเว็บไซต์ www.arduino.cc

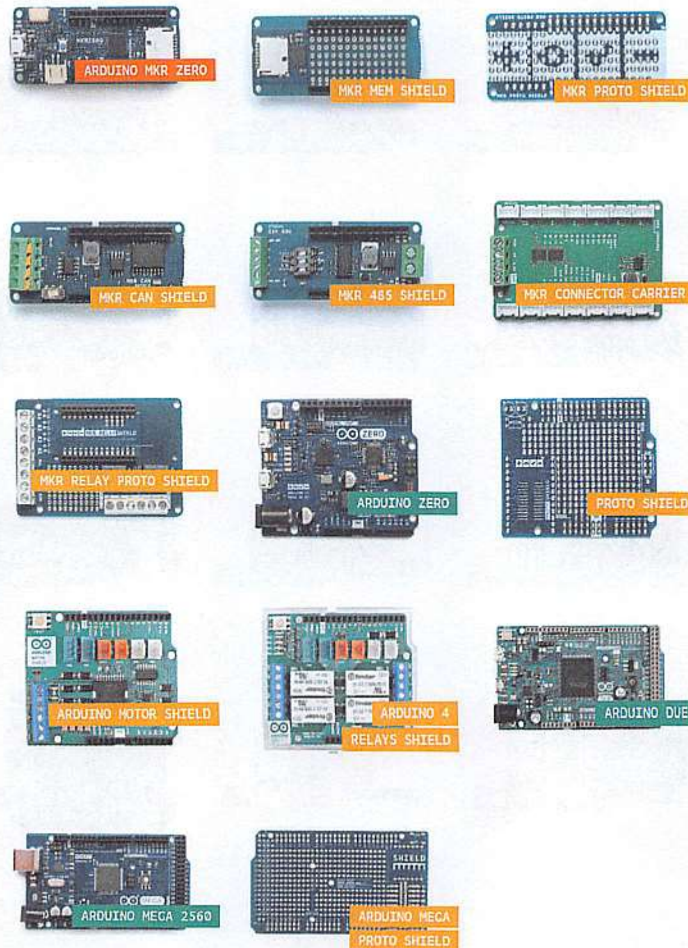
1. **Entry Level** เป็นกลุ่มสินค้าที่ออกแบบไว้สำหรับผู้เริ่มต้น เน้นใช้งานง่ายและเหมาะจะใช้เพื่อสร้างโปรเจ็กต์ชิ้นแรกสำหรับมือใหม่ที่กำลังก้าวสู่เมกเกอร์ (Maker) สินค้าในกลุ่มนี้ประกอบด้วยบอร์ดและโมดูลที่เหมาะในการเริ่มต้นเรียนรู้เกี่ยวกับบอร์ดอิเล็กทรอนิกส์ และการเขียนโค้ดมากที่สุด ในชุดการเรียนรู้ Arduino Starter Kit ก็จะแถมคู่มือที่ประกอบด้วย 15 บทเรียนที่จะสอนตั้งแต่เบสิกพื้นฐานไปจนถึงการทำโปรเจ็กต์ยากๆ กันเลยทีเดียว



▲ กลุ่ม Entry Level บอร์ดมาตรฐานรุ่นต่างๆ เหมาะสำหรับผู้เริ่มต้น

www.arduino.cc/en/Main/Products

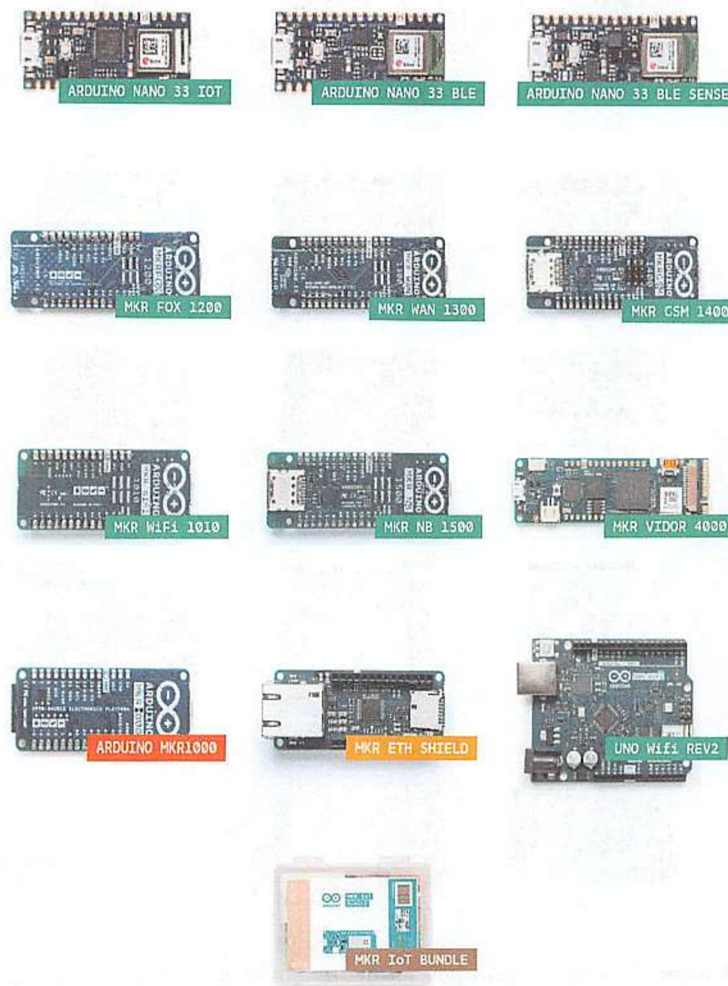
2. **Enhanced Features** เป็นการเพิ่มสเปคให้สูงขึ้นเพื่อรองรับโปรเจ็กต์ระดับสูงที่มีความซับซ้อนกว่ากลุ่ม Entry โดยเพิ่มทั้งฟังก์ชันการทำงาน และมีการทำงานที่รวดเร็วกว่าบอร์ดมาตรฐาน รวมถึงรองรับอุปกรณ์ I/O ได้มากขึ้นอีกด้วย



▲ กลุ่ม **Enhanced Features** บอร์ดที่มีฟังก์ชันขั้นสูงหรือทำงานได้เร็ว เหมาะสำหรับโปรเจ็กต์ที่ซับซ้อน

www.arduino.cc/en/Main/Products

3. **Internet of Things** เหมาะสำหรับงานที่ต้องใช้อินเทอร์เน็ตในการรับส่งข้อมูลและสั่งงาน เพื่อควบคุมอุปกรณ์ต่างๆ ให้ทำงานตามต้องการ โดยบอร์ดกลุ่มนี้รองรับ Arduino Cloud เพื่อใช้ในการรวบรวมข้อมูลลงในแดชบอร์ด การควบคุมอุปกรณ์จากแอปมือถือ และการอัพเดตเฟิร์มแวร์ทางเครือข่าย



▲ กลุ่ม IoT Products บอร์ดที่เหมาะกับงานที่ต้องการให้อุปกรณ์มีการเชื่อมต่อกันได้

www.arduino.cc/en/Main/Products

4. **Education** สินค้าในกลุ่มนี้จะเหมาะกับนักการศึกษา ผู้ที่ต้องการเครื่องมือทั้งฮาร์ดแวร์และซอฟต์แวร์ ที่จำเป็นต่อการสร้างประสบการณ์ตรงในการเรียนรู้เชิงนวัตกรรมมากขึ้น ชุด Education จะช่วยให้นักเรียนเกิดความสนุกสนาน และได้รับแรงบันดาลใจในโลกของการเขียนโปรแกรม และการประกอบชิ้นส่วนอิเล็กทรอนิกส์จนเป็นโครงงานสำเร็จรูป



▲ กลุ่ม Arduino Education เป็นบอร์ดพร้อมชุดประกอบสำเร็จที่ใช้งานได้จริง เหมาะจะใช้เพื่อการศึกษา
www.arduino.cc/en/Main/Products

นอกจากนี้ยังมีบอร์ดอีกมากมายที่ถูก Retired หรือยกเลิกการผลิตไปแล้ว ส่วนจะเป็นสินค้ารุ่นไหนบ้างนั้น เข้าไปดูได้ที่ www.arduino.cc/en/Main/Products

ตารางเปรียบเทียบสเปคบอร์ดรุ่นต่างๆ

Name	Processor	Operating/ Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V/ 7-12 V	32 MHz	6/0	14/4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V/ 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V ATmega328P	2.7-5.5 V/ 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V/ 2.7-5.5 V	8 MHz	4/0	9/4	1	2	32	-	-
LilyPad USB	ATmega32U4	3.3 V/ 3.8-5 V	8 MHz	4/0	9/4	1	2.5	32	Micro	-
Mega 2560	ATmega2560	5 V/7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V/7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V/5 V	48MHz	7/1	8/4	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V/ 3.35-12 V 5 V/5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V/ 3.35-12 V 5 V/5-12 V	8 MHz 16 MHz	6/0	14/6	1	2	32	-	1

Name	Processor	Operating/ Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
Uno	ATmega328P	5 V/7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V/ 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2
Due	ATSAM3X8E	3.3 V/ 7-12 V	84 MHz	12/2	54/12	-	96	512	2 Micro	4
Esplora	ATmega32U4	5 V/7-12 V	16 MHz	-	-	1	2.5	32	Micro	-
Ethernet	ATmega328P	5 V/7-12 V	16 MHz	6/0	14/4	1	2	32	Regular	-
Leonardo	ATmega32U4	5 V/7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
Mega ADK	ATmega2560	5 V/7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Mini	ATmega328P	5 V/7-9 V	16 MHz	8/0	14/6	1	2	32	-	-
Nano	ATmega168 ATmega328P	5 V/7-9 V	16 MHz	8/0	14/6	0.512 1	1 2	16 32	Mini	1
Yùn	ATmega32U4 AR9331 Linux	5 V	16 MHz 400 MHz	12/0	20/7	1	2.5 16 MB	32 64 MB	Micro	1
Arduino Robot	ATmega32u4	5 V	16 MHz	6/0	20/6	1 KB	2.5 KB	32 KB	1	1
MKRZero	SAMD21 Cortex-M0+ 32bit low power ARM MCU	3.3 V	48 MHz	7 /1	22/12	No	32 KB	256 KB	1	1

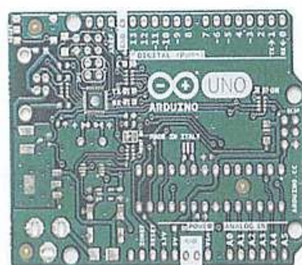
Arduino บอร์ดแท้บอร์ดเทียมดูกันอย่างไร

จากที่กล่าวไปแล้วในบทแรกว่า บอร์ด Arduino เป็นฮาร์ดแวร์โอเพ่นซอร์สที่เปิดเผยข้อมูลทางเทคนิคทุกอย่างให้ดาวน์โหลดกันไปดูได้ฟรีๆ ว่าบอร์ดมีการออกแบบลายวงจรแบบไหน ใช้อุปกรณ์อะไร ซอฟต์แวร์ที่ใช้ก็เปิดเผยด้วยเช่นกัน โดยอนุญาตให้ใครก็ได้นำไปพัฒนาต่อยอด จะผลิตขึ้นใช้เองหรือเพื่อวางจำหน่ายก็ไม่ขัดข้อง

บอร์ดมาตรฐานที่ผ่านการออกแบบและผลิตอย่างเป็นทางการโดย Arduino เอง จะมีโรงงานผลิตที่ประเทศอิตาลี สังเกตได้จากใต้บอร์ดจะมีตัวหนังสือสกรีนบนบอร์ดว่า "Made in Italy" เราจึงเรียกว่า "Official Board" ส่วนบอร์ดที่ผลิตโดยผู้นำไปพัฒนาต่อจะเรียกว่า "Compatible Board" ซึ่งจะปรับแต่งหรือเปลี่ยนแปลงอุปกรณ์บางส่วน เพื่อตอบสนองความต้องการของกลุ่มเป้าหมายที่เฉพาะเจาะจงลงไป โดยมากจะใช้อุปกรณ์และคุณภาพการผลิตที่มีมาตรฐานต่ำกว่าเพื่อขายถูกกว่า บอร์ดเหล่านี้จะทำงานได้ใกล้เคียงกับบอร์ด Arduino จึงสามารถซื้อมาใช้งานทดแทนกันได้ และเนื่องจากบอร์ด Compatible Board มักผลิตขึ้นในประเทศจีน จึงทำให้มีราคาที่ถูกกว่า Official Board จึงเป็นทางเลือกที่ดีสำหรับผู้ที่มีงบประมาณน้อย

ในหัวข้อนี้จะแสดงวิธีสังเกตว่า บอร์ดแท้แตกต่างจากบอร์ดเทียมอย่างไร

1. **Color** บอร์ด Arduino ของแท้ตัวบอร์ดจะมีสีเขียวนกเป็ดน้ำ ซึ่งถ้าเราสังเกตดีๆ จะเห็นว่ามี 2 สีผสมกัน คือ สีเขียวและสีน้ำเงิน แต่ถ้าเป็นบอร์ดเทียมจะมีสีน้ำเงินและน้ำเงินเข้ม



บอร์ดแท้
Original Board



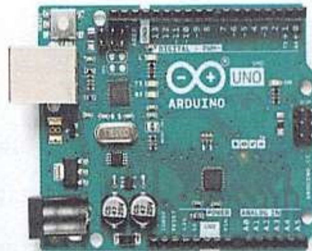
บอร์ดเทียม
Counterfeit



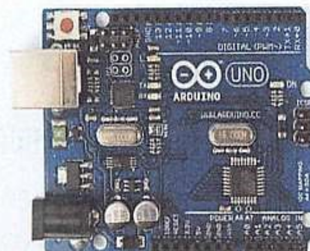
บอร์ดแท้ Arduino UNO Rev3
Original Board



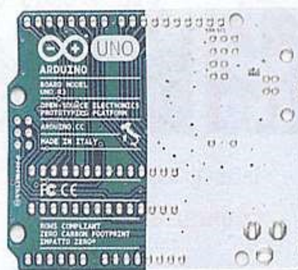
บอร์ดเทียม Arduino UNO Rev3
Counterfeit



บอร์ดแท้ Arduino UNO SMD
Original Board



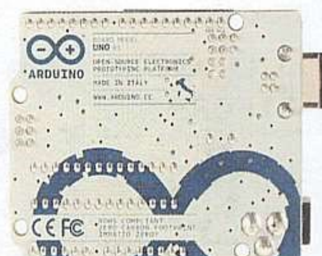
บอร์ดเทียม Arduino UNO SMD
Counterfeit



บอร์ดแท้
Original Board



บอร์ดเทียม
Counterfeit



บอร์ดแท้
Original Board



บอร์ดเทียม
Counterfeit

2. **Font/Silk** ถ้าสังเกตให้ดีจะพบว่า ฟอนต์ของคำว่า 'UNO' หรือ 'Arduino' จะเขียนไม่เหมือนกัน โดยของ Arduino จะใช้ฟอนต์ที่ได้รับการออกแบบมาโดยเฉพาะ ซึ่งทางผู้ผลิตบอร์ดเทียมไม่สามารถลอกเลียนให้เหมือนกันได้ สังเกตตัว 'O' จะแตกต่างกันอย่างชัดเจน





บอร์ดแท้ **Original Board** บอร์ดเทียม **Counterfeit**

3. **Arduino Logo** บอร์ดแท้จะใช้โลโก้ที่มีลักษณะแบบนี้แบบเดียวเท่านั้น ในขณะที่บอร์ดเทียมจะใช้ฟอนต์คนละตัวกับบอร์ดแท้ สังเกตตัว 'A' ของแท้จะมีช่องและขอบของฟอนต์จะมีเหลี่ยม ในขณะที่บอร์ดเทียมตัว 'A' จะไม่มีช่องและขอบของฟอนต์กลมมน หรือบางแห่งโลโก้จะมีลักษณะเอียง ดังรูป







บอร์ดแท้ **Original Board** บอร์ดเทียม **Counterfeit**

4. **Made in Italy** เมื่อพลิกไปดูที่ด้านหลังของบอร์ด จะเห็นแผนที่ประเทศอิตาลีเพื่อเป็นการแสดงความเคารพต่อถิ่นกำเนิด (หมายถึง สถานที่ผลิต) แผนที่ที่สกรีนลงบนบอร์ดแท้จะคมชัด สีหมึกสกรีนมีน้ำหนักเท่ากัน และมีรูปร่างที่สวยงามกว่าบอร์ดเทียม









บอร์ดแท้ **Original Board** บอร์ดเทียม **Counterfeit**

5. **Connections and Connectors** เปรียบเทียบการเดินลายวงจร หรืออาจจะเรียกว่า 'ลายพรินต์' บนแผ่น PCB (Print Circuit Board) ซึ่งเป็นทางเดินสัญญาณไฟฟ้าไปสู่ชิ้นส่วนอิเล็กทรอนิกส์ที่อยู่บนแผงวงจร ถ้าเป็นบอร์ดแท้ลายวงจรจะดูได้มาตรฐานสวยงาม ในขณะที่ของเทียมจะดูไม่สวยนัก เห็นถึงมาตรฐานการผลิตที่มีคุณภาพอย่างชัดเจน



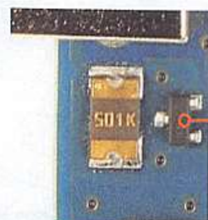
บอร์ดแท้
Original Board



บอร์ดเทียม
Counterfeit

6. **Keywords** ในขณะที่เลือกซื้อออนไลน์ลองใช้คำค้นหา เช่น 'arduino compatible' หรือ 'Uno for Arduino' หากบอร์ดดูคล้าย Original Arduino ส่วนใหญ่จะเป็นบอร์ดเทียม

7. **Components** วิธีดูของเทียมที่สำคัญ จะพบอุปกรณ์อยู่ชิ้นหนึ่งจะมีโค้ด 501K ที่ตำแหน่งใกล้กับ Voltage Regulator หากมีสีเขียวแสดงว่าเป็นบอร์ดเทียม สำหรับบอร์ดแท้จะมีสีดำทอง ดังรูป ซึ่งทาง Arduino จงใจทำให้มีลักษณะพิเศษเช่นนี้ไว้ ก็เพราะต้องการให้ Arduino นั้นดูแตกต่าง



บอร์ดแท้
Original Board



บอร์ดเทียม
Counterfeit

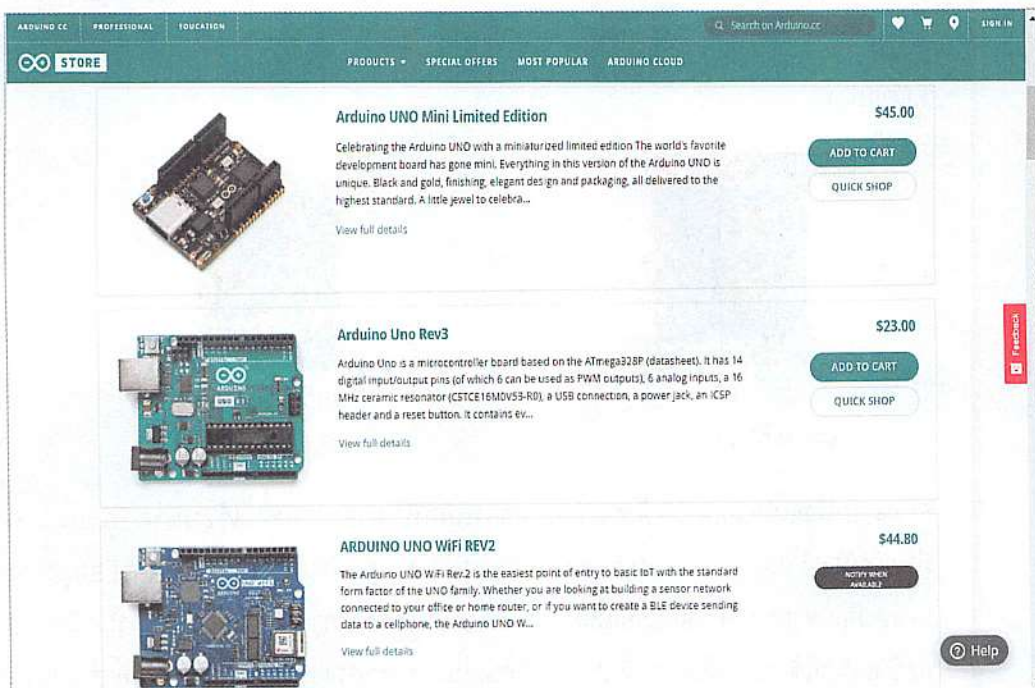
8. **Price** บอร์ดแท้จะถูกผลิตด้วยกระบวนการผลิตที่มีมาตรฐานสูง ใช้อุปกรณ์ที่มีคุณภาพ มีบรรจุภัณฑ์ และคู่มือที่การันตีความพอใจ สรุปว่างานละเอียดกว่ากันมาก ราคาย่อมสูงกว่าบอร์ดเทียมจากประเทศจีนหรือแหล่งอื่นๆ หากคำนึงถึงอายุการใช้งานก็ขอแนะนำให้เลือกซื้อบอร์ดแท้มาใช้ แต่ถ้าต้องการซื้อมาเพื่อศึกษาในงบประมาณจำกัด ก็สามารถเลือกซื้อบอร์ด Arduino Compatible มาทดแทนได้ **Credit :** www.arduino.cc/en/Products/Counterfeit

แหล่งเลือกซื้อบอร์ด Arduino และ Sensors

แหล่งซื้อบอร์ดและอุปกรณ์ต่างๆ ของ Arduino สามารถซื้อได้ตามร้านค้าทั้งแบบ Online และแบบ Offline ทั้งในและต่างประเทศ ราคาจะแตกต่างกันตามแหล่งที่ซื้อ จำนวน และค่าขนส่ง แต่ในที่นี้จะขอแนะนำการเลือกซื้อผ่านทางร้านค้าออนไลน์

เลือกซื้อผ่านเว็บ arduino.cc

เราสามารถเลือกซื้อของแท้กับผู้ผลิตโดยตรงได้ที่เว็บไซต์ www.arduino.cc (Arduino Official) รับประกันว่าได้ของแท้ Made in Italy แต่ก็มีราคาแพงกว่าที่ขายกันตามท้องตลาด เนื่องจาก Arduino เป็น Open Source Hardware ที่ให้ผู้ผลิตรายอื่นสามารถผลิตขายได้ในราคาถูก แต่สินค้าก็มีคุณภาพที่แตกต่างกัน แม้ว่าจะสามารถทำงานได้พอๆ กันก็ตาม (เช่น Made in China) โดยของแท้จะมีอายุการใช้งานที่มากกว่า และเนื่องจากต้องส่งมาจากต่างประเทศจึงจะใช้เวลานานกว่า หรือสามารถซื้อผ่านร้านค้าในไทยที่นำเข้ามาได้เช่นกัน แต่ราคาอาจจะแพงกว่าหน้าเว็บเล็กน้อยขึ้นอยู่กับจำนวนการสั่งซื้อ



▲ รูปแสดงหน้าเว็บเพจร้านค้าออนไลน์

https://store.arduino.cc/usa/

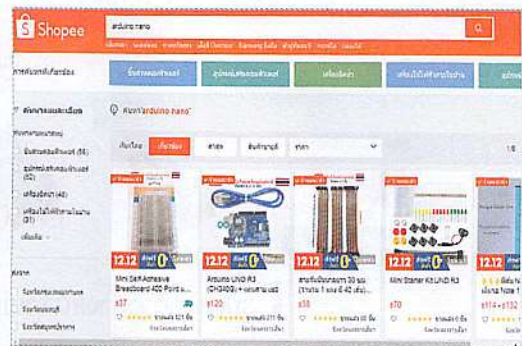
เลือกซื้อใน Lazada และ Shopee

Market Place แหล่งรวมร้านค้าออนไลน์ขนาดใหญ่ที่มีเครื่องมืออำนวยความสะดวกให้กับทั้งผู้ซื้อและผู้ขาย และเป็นที่รู้จักอย่างกว้างขวาง คือ เว็บไซต์ Lazada และ Shopee ที่หยิบยกทั้ง 2 เจ้ามาแนะนำก็เพราะคุณภาพในเรื่องของ UX/UI หมายถึง หน้าเว็บเพจมีการออกแบบที่สอดคล้องกับการใช้งาน ช่วยสร้างประสบการณ์ที่ดีในการ Shopping ซึ่งทำได้ดีกว่าเว็บไซต์ข้างนอก เช่น เห็นสินค้าจากหลายร้านพร้อมๆ กัน เปรียบเทียบราคาได้ทันที มีรีวิวจากลูกค้าประกอบการตัดสินใจ เป็นต้น



▲ รูปแสดงหน้าเว็บเพจ Lazada เมื่อค้นหาด้วยคำว่า 'arduino uno r3'

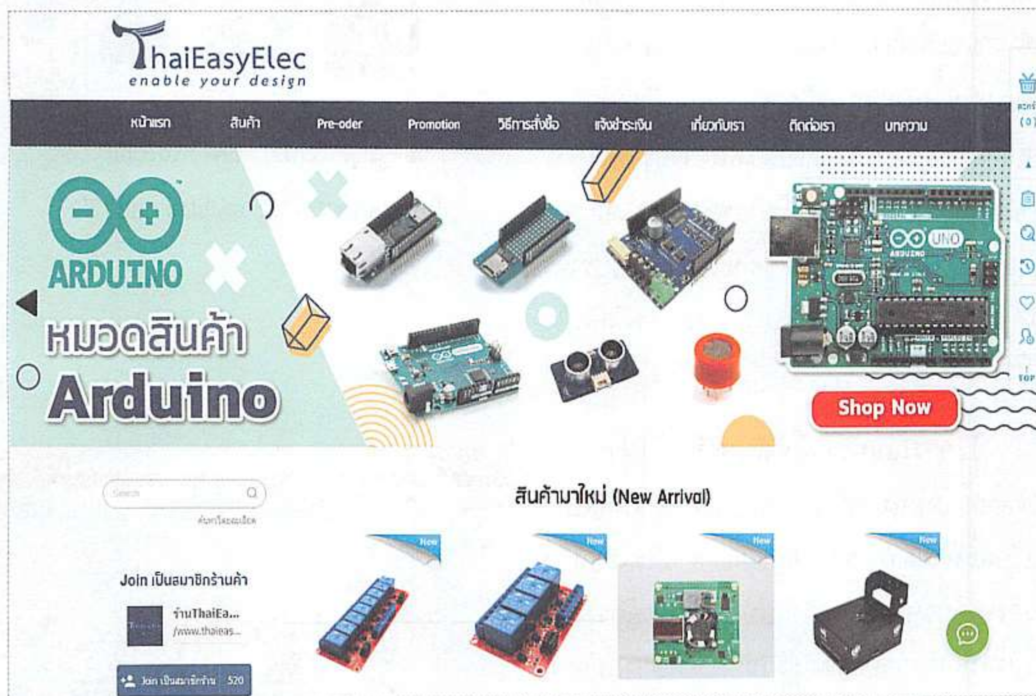
วิธีการเลือกซื้อ ให้ค้นหาสินค้าผ่านช่อง Search โดยพิมพ์คำว่า 'Arduino' + รุ่นที่สนใจ หรือสินค้าต่างๆ ของ Arduino จะมีให้เลือกทั้งบอร์ดเดี่ยวๆ หรือจะสั่งแบบจัดเป็นชุดโครงงานก็มีให้เลือก ราคาแต่ละร้านจะแตกต่างกันตามโปรโมชัน ดังนั้น ควรเปรียบเทียบราคา เรทติ้ง การรับประกัน ระยะเวลาการส่ง และตรวจสอบคุณภาพสินค้าผ่านรีวิวจากผู้ซื้อก่อนหน้า เพื่อประกอบการตัดสินใจ สามารถเลือกชำระเงินแบบเก็บเงินปลายทาง หรือชำระเงินแบบออนไลน์ก็ได้



▲ รูปแสดงหน้าเว็บเพจ Shopee เมื่อค้นหาด้วยคำว่า 'arduino nano'

เลือกซื้อผ่านร้านค้าออนไลน์

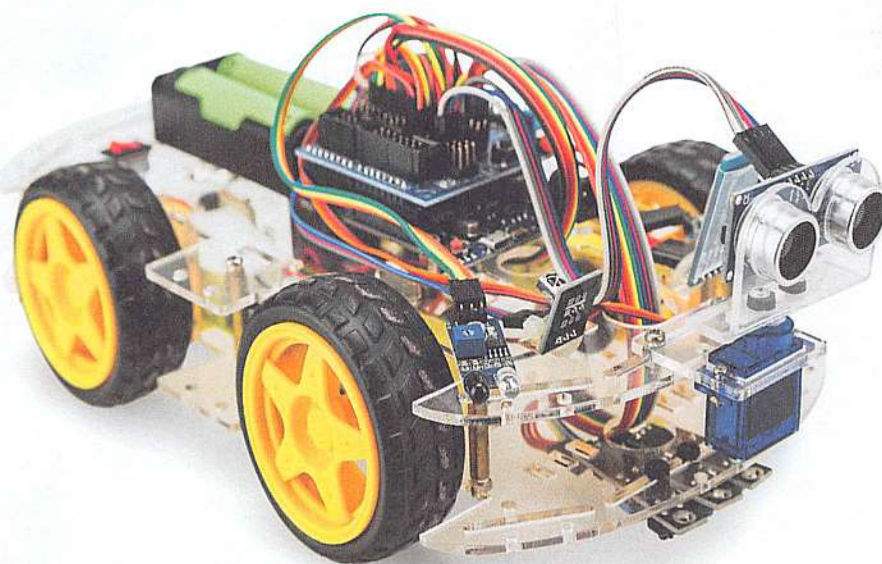
ร้านค้าอุปกรณ์อิเล็กทรอนิกส์ออนไลน์ที่มีเว็บไซต์ของตนเองนั้น มีให้เลือกมากมาย สามารถค้นหาได้ทาง www.google.com โดยพิมพ์คำว่า "ร้านขาย arduino" ก็จะมีขึ้นอยู่หลายร้าน เช่น arduitronics, thaieasyelec, micontechlab, arduinothai, arduinoall เป็นต้น นอกจากร้านค้าเหล่านี้จะจำหน่ายสินค้า Arduino แล้วยังมีบอร์ดยี่ห้ออื่นๆ ให้เลือกซื้ออีกด้วย



▲ รูปแสดงตัวอย่างหน้าเว็บเพจของร้าน THAIEASYELEC

บทสรุปท้ายบท

จากบทที่ 2 เราได้ศึกษาบอร์ด Arduino ในทางฮาร์ดแวร์ รู้จักส่วนประกอบต่างๆ บนบอร์ด รวมถึงอุปกรณ์เสริมที่ทำงานร่วมกับบอร์ดหลัก และในบทที่ 3 นี้เป็นคำแนะนำในการเลือกซื้อบอร์ด Arduino ให้เหมาะกับงาน หรือตรงตามความต้องการในการนำไปใช้งาน ซึ่งต้องอาศัยความรู้พื้นฐานจากบทที่ 2 และบทที่ 3 ร่วมกัน เพราะในการพิจารณาเลือกซื้อนั้น จำเป็นจะต้องอ่านสเปคได้ ดูฟอร์มแฟ็กเตอร์ของบอร์ดเป็น (รูปร่างลักษณะเผื่อกรณีจะต้องทำเคสที่มีขนาดเหมาะสม) รู้ว่าจะไปหาซื้อของถูกของดีของแท้ได้จากที่ไหน และที่สำคัญยังแยกแยะได้ว่า บอร์ดแท้ต่างจากบอร์ดเทียมอย่างไร บทต่อไปจะเป็นพาร์ตของ Software & Programming ซอฟต์แวร์เพื่อการเรียนรู้และเขียนโปรแกรม Arduino



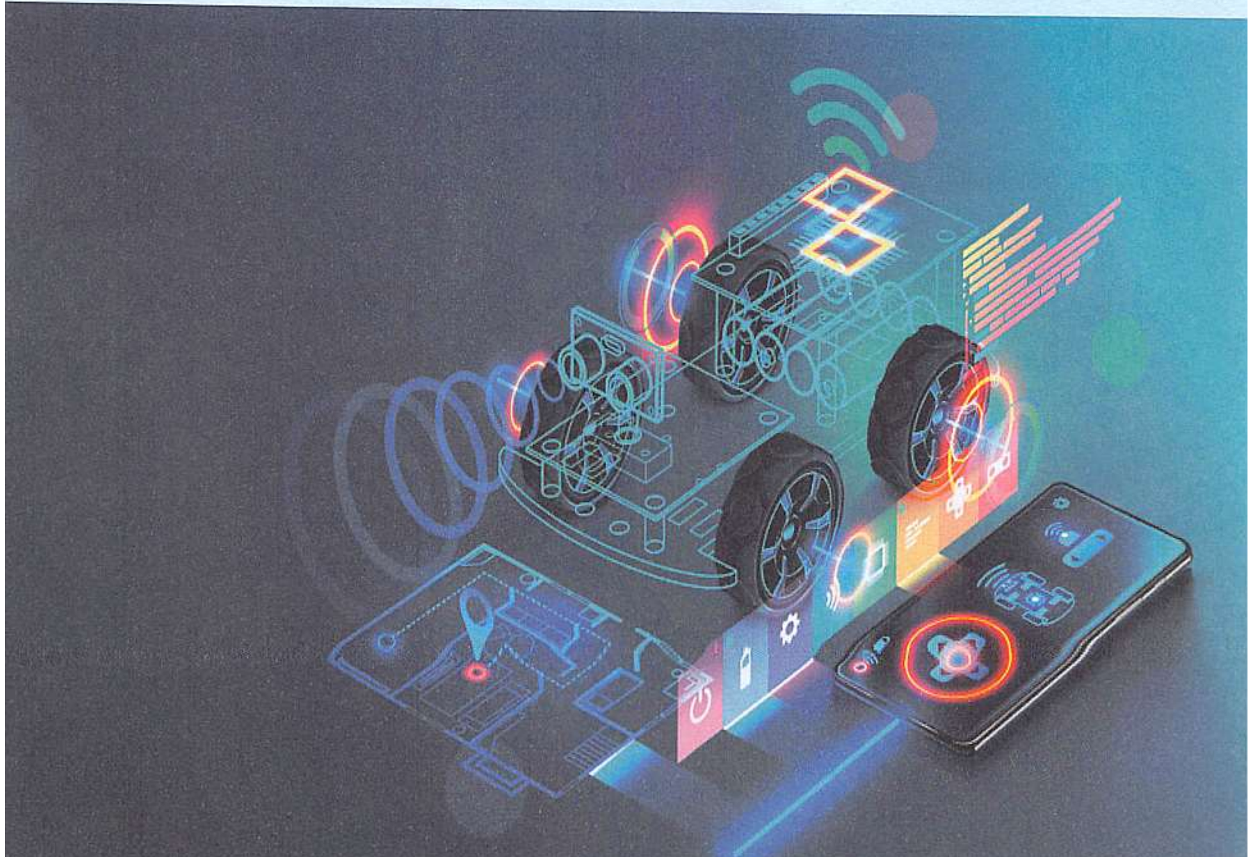
CHAPTER

04

ซอฟต์แวร์ เพื่อการเรียนรู้ Arduino

Circuit Drawing and Simulation

ในบทที่ 1 ถึง 3 เราได้ทำความรู้จักกับบอร์ด Arduino และอุปกรณ์เสริมต่างๆ ซึ่งเป็นเรื่องทางฮาร์ดแวร์ทั้งสิ้น สำหรับบทที่ 4 จะอธิบายในส่วนของซอฟต์แวร์และการเขียนโปรแกรม จากบทแรกเราได้เกริ่นนำซอฟต์แวร์สำหรับเขียนโค้ดกันไปบ้างแล้ว ซึ่งจะได้สรุปหลักการเขียนด้วยภาษา C กันในบทที่ 6 สำหรับบทที่ 4 จะแนะนำซอฟต์แวร์อีก 2 ส่วน ได้แก่ ซอฟต์แวร์สำหรับการเขียนวงจร (Circuit Drawing) และซอฟต์แวร์จำลองการทำงานแทนบอร์ดจริง (Arduino Simulator) โดยจะเน้นแสดงตัวอย่างเบื้องต้นแค่พอเข้าใจเท่านั้น เพื่อเป็นแนวทางให้ผู้อ่านนำไปศึกษาต่อด้วยตนเอง

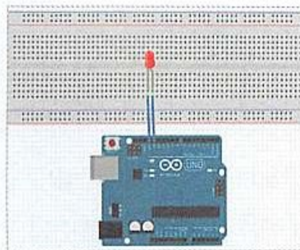


CHAPTER | 04

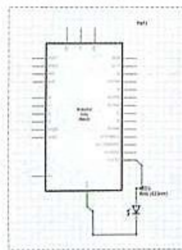
Fritzing (Circuit Drawing)

"เหตุผลที่แนะนำซอฟต์แวร์ตัวนี้เป็นตัวแรกก็เพราะว่า เว็บไซต์ Arduino แนะนำให้ใช้"

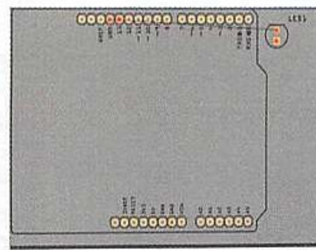
Fritzing คือ โปรแกรมที่ใช้จำลองการต่อบอร์ด Arduino เข้ากับอุปกรณ์อิเล็กทรอนิกส์โดยใช้ภาพกราฟิกแทนของจริง สามารถใช้เขียนแบบวงจรไฟฟ้า และออกแบบลายวงจรบนแผ่น PCB ได้อีกด้วย จึงสามารถแสดงวงจรในรูปแบบ Breadboard, Schematic และ PCB ได้ นอกจากนี้ยังสามารถใช้ออกแบบแก้ไข และสร้างชิ้นส่วน (Parts) ของอุปกรณ์อิเล็กทรอนิกส์ขึ้นได้เองอีกด้วย



Breadboard



Schematic



PCB

▲ รูปแสดงวงจรในรูปแบบต่างๆ ที่สร้างจากโปรแกรม Fritzing

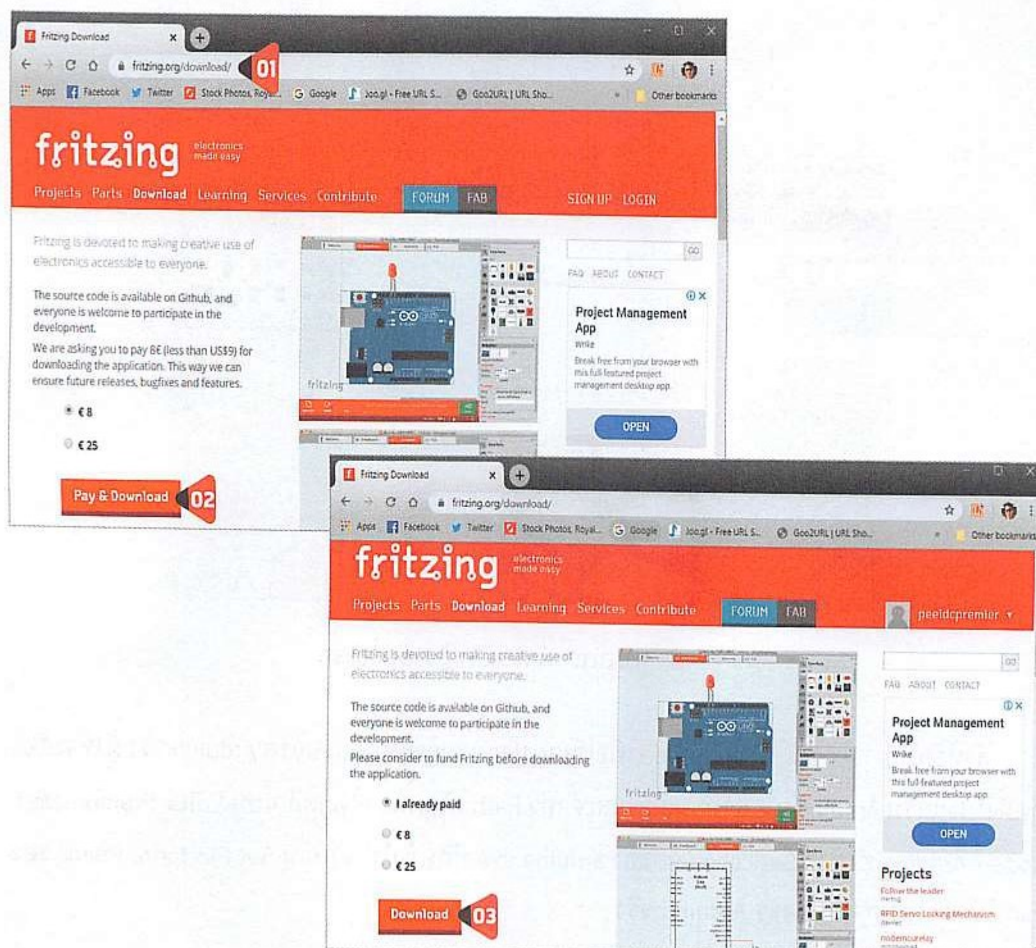
จุดเด่นของโปรแกรม Fritzing

1. มีบอร์ด Arduino และอุปกรณ์ที่เกี่ยวข้องมากมาย รองรับการเขียนวงจรหลากหลายรูปแบบ
2. สามารถใช้โปรแกรมนี้เพื่อเขียนโค้ดและอัพโหลดลงบอร์ด Arduino ได้เช่นเดียวกับ Arduino IDE เลยทีเดียว
3. ได้รับการพัฒนาเครื่องมือให้ยูสเซอร์ใช้ทำเอกสาร Arduino และเอกสารทางอิเล็กทรอนิกส์ทั้งหลาย
4. สามารถแชร์เอกสารกับคนอื่นๆ ได้ง่าย
5. รองรับการสอนวิชาอิเล็กทรอนิกส์ในห้องเรียน
6. ใช้สร้าง PCB Layout เพื่อใช้ในการผลิตด้วยเครื่องจักร
7. สนับสนุนฮาร์ดแวร์โอเพ่นซอร์ส หมายถึง การเปิดเผยข้อมูลเบื้องหลังการสร้างและพัฒนา

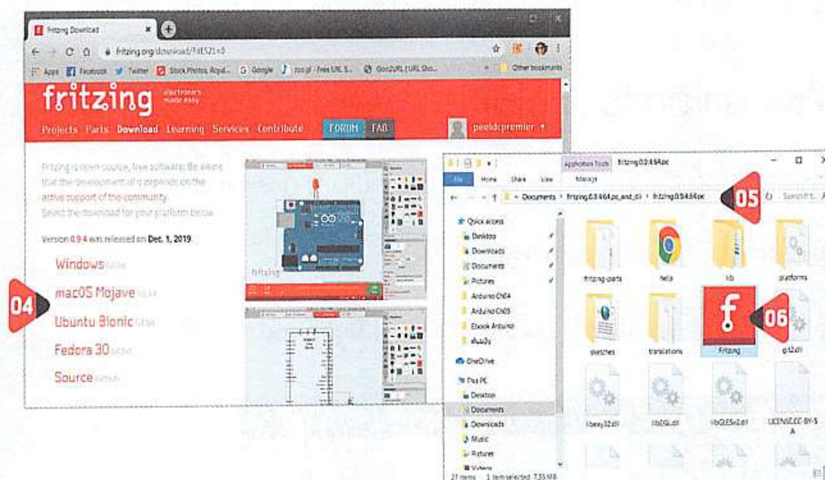
8. โปรแกรมได้รับการออกแบบมาสำหรับคนในหลากหลายอาชีพ ไม่เว้นแม้กระทั่งมือสมัครเล่น
9. มีชุมชนออนไลน์เพื่อใช้เป็นแหล่งแลกเปลี่ยนความรู้และแชร์ประสบการณ์ (บอร์ดฟอรัม : https://forum.fritzing.org/)
10. มีบทเรียนสอนการใช้งานโปรแกรมบนเว็บไซต์ (https://fritzing.org/learning/)

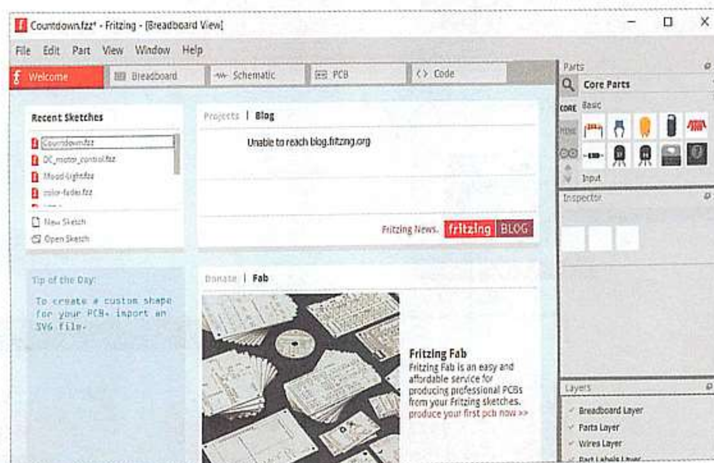
ดาวน์โหลดและติดตั้ง Fritzing

1. ให้เข้าไปที่เว็บไซต์ http://fritzing.org และคลิกเมนู **Download**
2. สมัครสมาชิก **Sign Up** และ **Login** เข้าใช้งาน
3. คลิกเลือกออปชัน **I already paid** และคลิกปุ่ม **Download**



4. เลือกดาวน์โหลดตามระบบปฏิบัติการของเครื่องที่ใช้งาน
5. เมื่อดาวน์โหลดเสร็จแล้ว ให้เข้าไปที่ตำแหน่งเก็บไฟล์ดาวน์โหลด แล้วทำการ Unzip
6. ดับเบิลคลิกไฟล์ชื่อ Fritzing เพื่อเปิดใช้งานโปรแกรม





▲ รูปแสดงหน้าจอเปิดใช้งานครั้งแรก

สำหรับในการใช้งานในโหมด Code เพื่อเขียนโปรแกรมอัพโหลดลงบอร์ด Arduino จริง ผู้ใช้จะต้องติดตั้งโปรแกรม Arduino IDE ไว้ก่อนแล้ว และตั้งค่า Path ให้ถูกต้อง โดยคลิกเมนู Edit > Preferences > Code > Arduino ทำการ Set Location ของ arduino.exe ที่ติดตั้งไป จากนั้น Set Platform, Board และ Port ให้ตรงตามการต่อวงจร Arduino จริง

การเขียนวงจรและไดอะแกรมการเดินสายไฟด้วย Fritzing

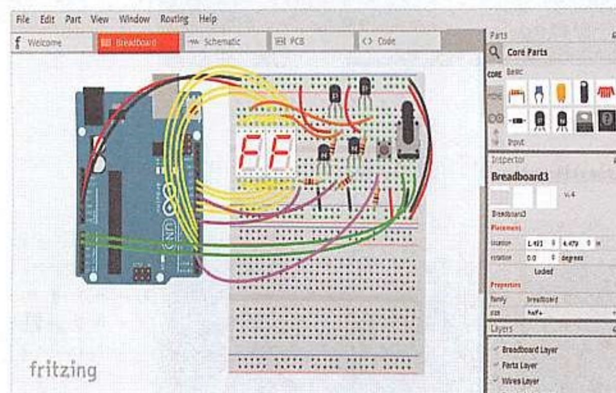
(Making Circuit & Wiring Diagrams)

เมื่อเราเปิดเรียกใช้งานโปรแกรม Fritzing หน้าจอ Welcome จะปรากฏขึ้นเป็นหน้าจอแรก ด้านบนสุดจะแสดงเมนูคำสั่งเสมอเหมือนกับโปรแกรมทั่วๆ ไป ถัดลงมาจะเป็นแถบสำหรับเลือก View หน้าเพจการทำงาน ได้แก่

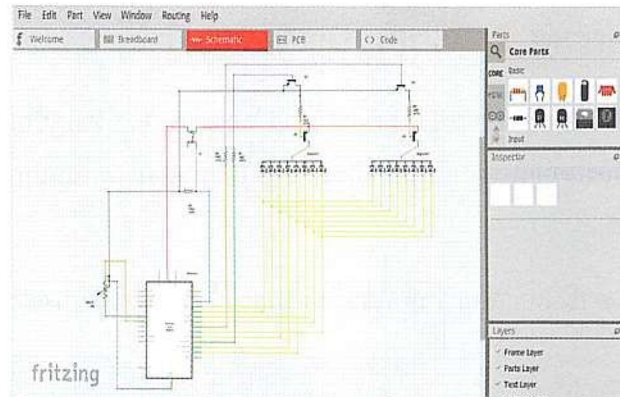
1. Welcome View เป็นหน้าจอแรกเพื่อใช้สร้าง Sketch ใหม่ หรือเปิด Sketch เก่า



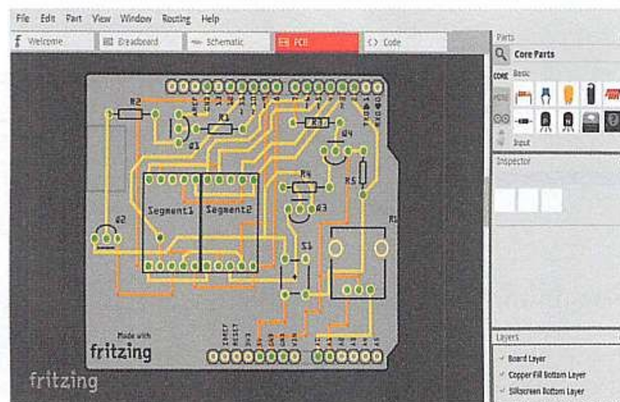
2. Breadboard View เป็นหน้าจอสำหรับเขียนวงจรในแบบจำลองบนบอร์ดทดลองด้วยภาพกราฟิก



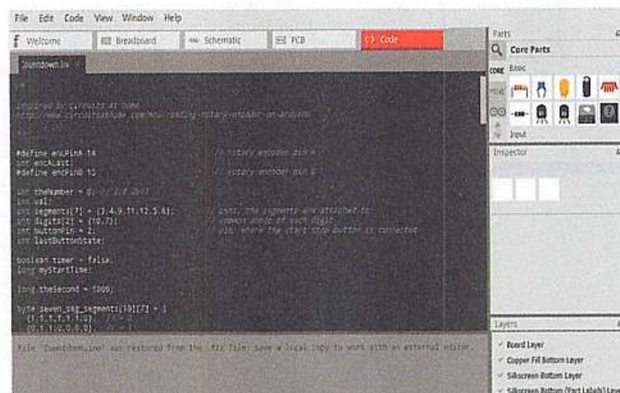
3. Schematic View เป็นหน้าจอสำหรับเขียนแผนผังหรือไดอะแกรมวงจร



4. PCB View เป็นหน้าจอสำหรับเขียนลายวงจรบนแผ่นพรินต์

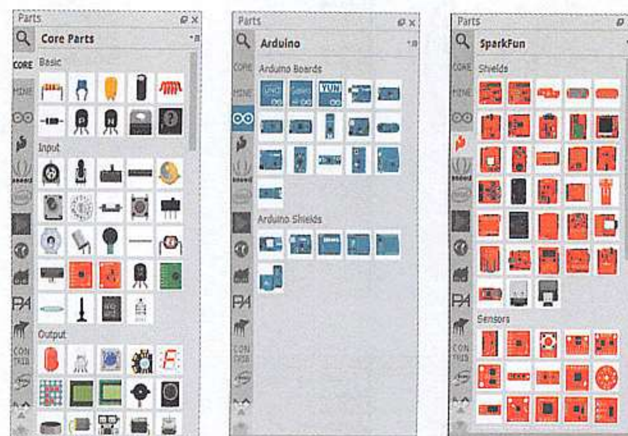


5. Code View เป็นหน้าจอสำหรับเขียนโค้ดเพื่อควบคุมการทำงาน



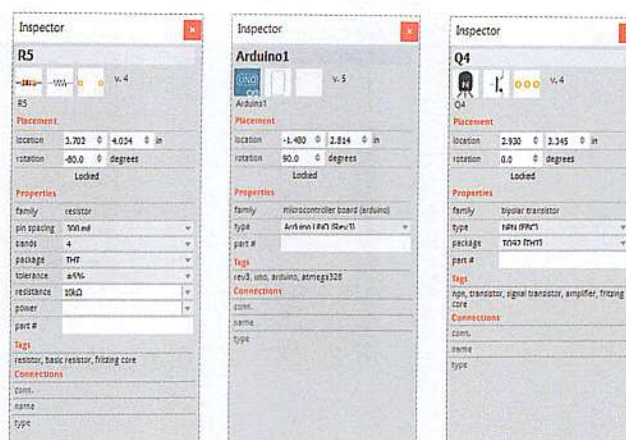
ส่วนทางด้านขวาจะแบ่งพื้นที่การทำงานออกเป็น 3 ส่วน ได้แก่

- **Parts** ชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์ ซึ่งจะมีการเก็บไว้เป็นหมวดหมู่เพื่อสะดวกในการค้นหา เช่น CORE ชิ้นส่วนอุปกรณ์หลัก, MINE ชิ้นส่วนที่เราสร้างขึ้นใหม่ (เริ่มต้นจะยังไม่มี), Arduino รวม Board และ Shields เป็นต้น ให้ผู้อ่านลองคลิกกลุ่มอื่นๆ เพื่อสำรวจอุปกรณ์ในแต่ละกลุ่ม



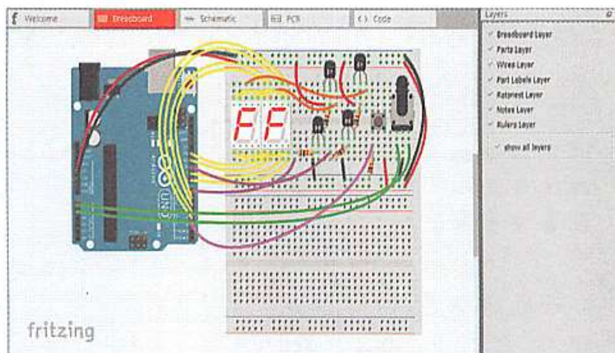
▲ รูปแสดงส่วนของ Parts ในกลุ่ม Core, Arduino และ SparkFun ตามลำดับ

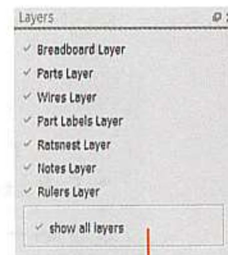
- **Inspector** เป็นส่วนแสดง Properties/Option/Value ของ Parts ที่ใช้ เราสามารถปรับแต่งค่าได้ที่นี่ สังเกตว่ารายละเอียดของ Inspector จะเปลี่ยนไปตาม Parts ที่เราเลือก โดยจะแสดงคุณสมบัติที่มีความสัมพันธ์กับ Parts ที่เราเลือก



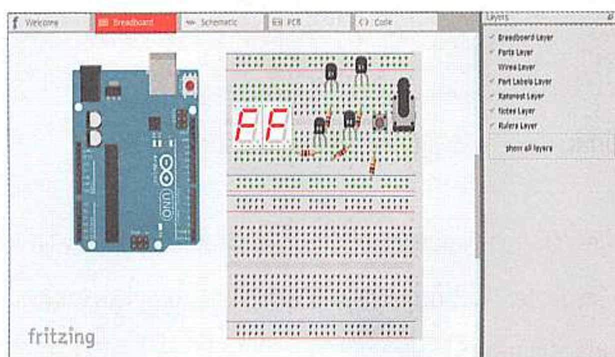
▲ รูปแสดงส่วนของ Inspector ของ Resistor, Arduino และ Transistor

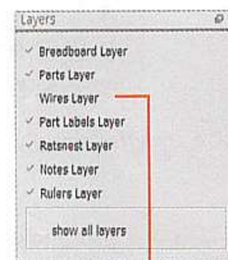
- **Layers** ลำดับชั้นของการซ้อนทับกันของ Parts เราสามารถปิดการมองเห็น Layers ต่างๆ ได้ตามความต้องการ ทั้งนี้เพื่อโฟกัสในสิ่งที่เราสนใจ หรือต้องการเคลียร์สิ่งที่ไม่ยังไม่สนใจออกไปก่อนชั่วคราว หรือใช้สร้าง Document ที่มีรายละเอียดแตกต่างกัน



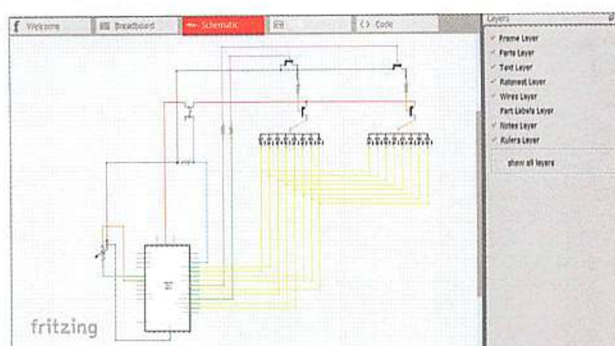


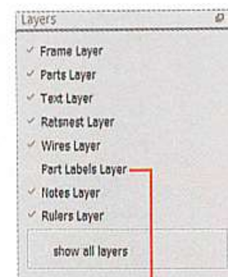
โชว์หมดทุก Layers





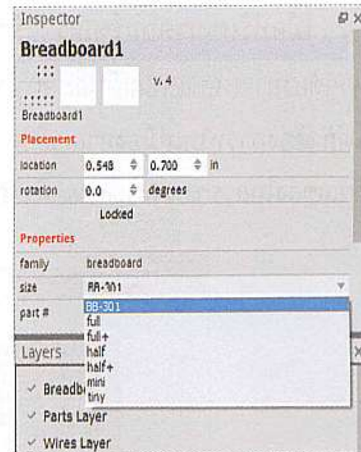
ปิดการมองเห็น Wires Layer



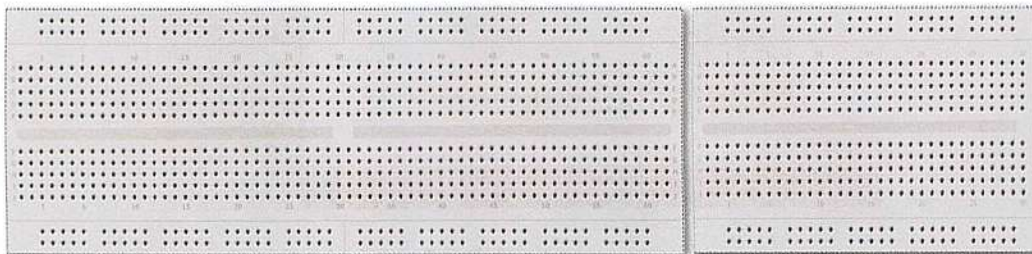


ปิดการมองเห็น Part Labels Layer

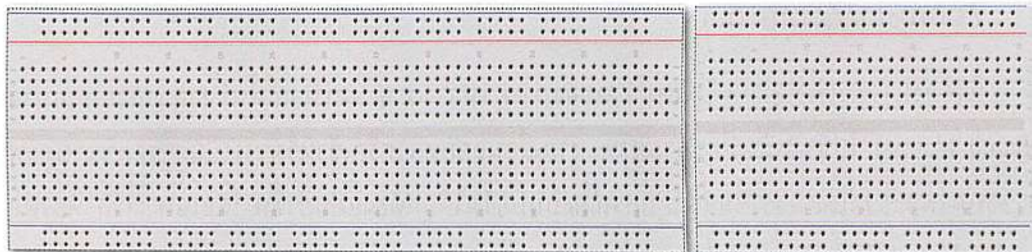
การปรับขนาด-หมุน Breadboard



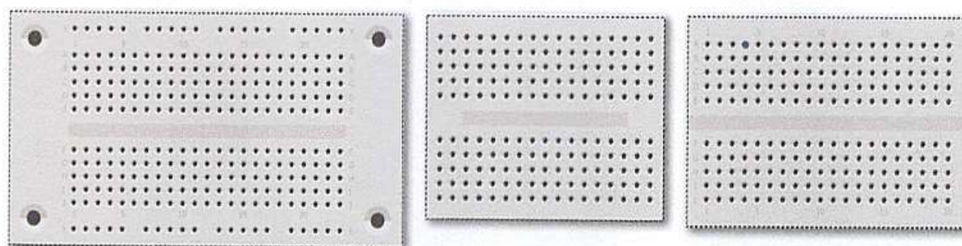
เริ่มต้นด้วยการรู้จักวิธีจัดการกับ Breadboard ซึ่งเป็นการจำลองภาพของบอร์ดทดลองขึ้นมา ซึ่งในส่วนนี้จะเป็นพื้นที่สำหรับการเขียนวงจรเพื่อสร้าง Document ขึ้นมา สิ่งที่ต้องเรียนรู้เป็นลำดับแรก คือ การเลือกขนาดของ Breadboard ให้เหมาะสมกับขนาดของวงจร ซึ่งเราสามารถกำหนดขนาดได้ที่ **Inspector** ที่หัวข้อหลัก **Properties** หัวข้อย่อย **size** เมื่อคลิกปุ่ม **Dropdown** จะพบตัวเลือกขนาดบอร์ดที่แตกต่างกัน ดังนี้



full half

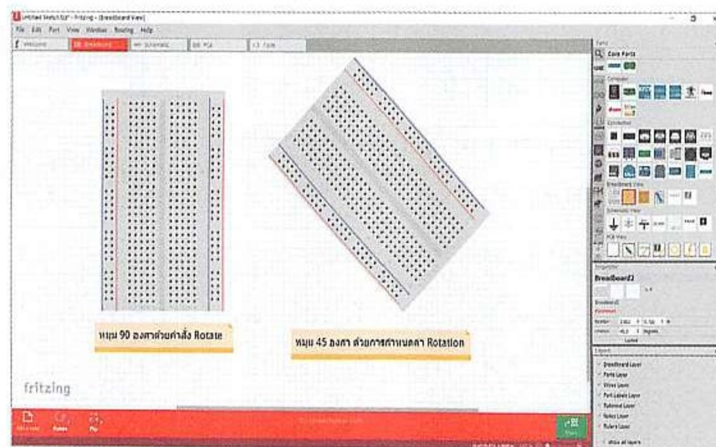


full+ half+



BB-301 mini tiny

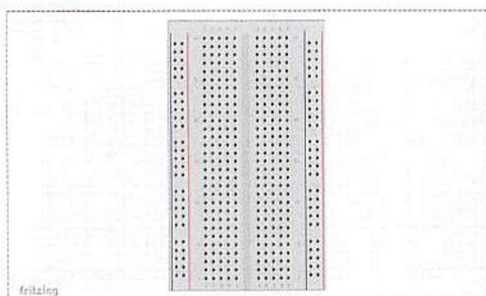
เมื่อปรับขนาดบอร์ดเสร็จแล้ว เราสามารถปรับเปลี่ยนมุมมองโดยการหมุนบอร์ดได้อีกด้วย หากต้องการหมุนแบบสเต็ปละ 90 องศา ก็สามารถคลิกที่ Rotate ใต้ภาพได้เลย (สลับด้านก็คลิกที่ Flip) แต่ถ้าต้องการหมุนที่มีองศาละเอียดกว่า 90 เช่น 30 องศา, 45 องศา, 90.5 องศา ก็สามารถระบุเป็นตัวเลขลงในส่วนของ Inspector หัวข้อ rotation ของ Properties (ใช้เครื่องหมาย – เพื่อหมุนกลับทิศทาง)



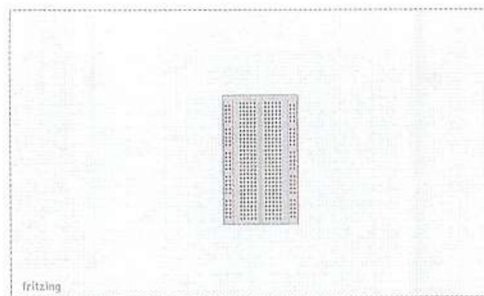
▲ รูปแสดงการหมุนบอร์ด หรือ Rotate

Zoom-in/Zoom-out

ถ้าต้องการซูมเพื่อขยายหรือลดขนาดของภาพ ให้วางเมาส์ในบริเวณพื้นที่งาน แล้วเลื่อนเมาส์ (Scroll Mouse) ไปข้างหน้าเพื่อขยายภาพ หากเลื่อนถอยหลังหรือเข้าหาตัวจะเป็นการลดขนาดภาพ การซูมภาพจะช่วยให้เราวางชิ้นส่วนอุปกรณ์ลงบนบอร์ดได้ถูกต้องแม่นยำขึ้น โดยเฉพาะเวลาต้องต่อสายไฟเข้ากับขาสัญญาณบนบอร์ด Arduino



เลื่อนเมาส์ไปข้างหน้าเพื่อ Zoom-in



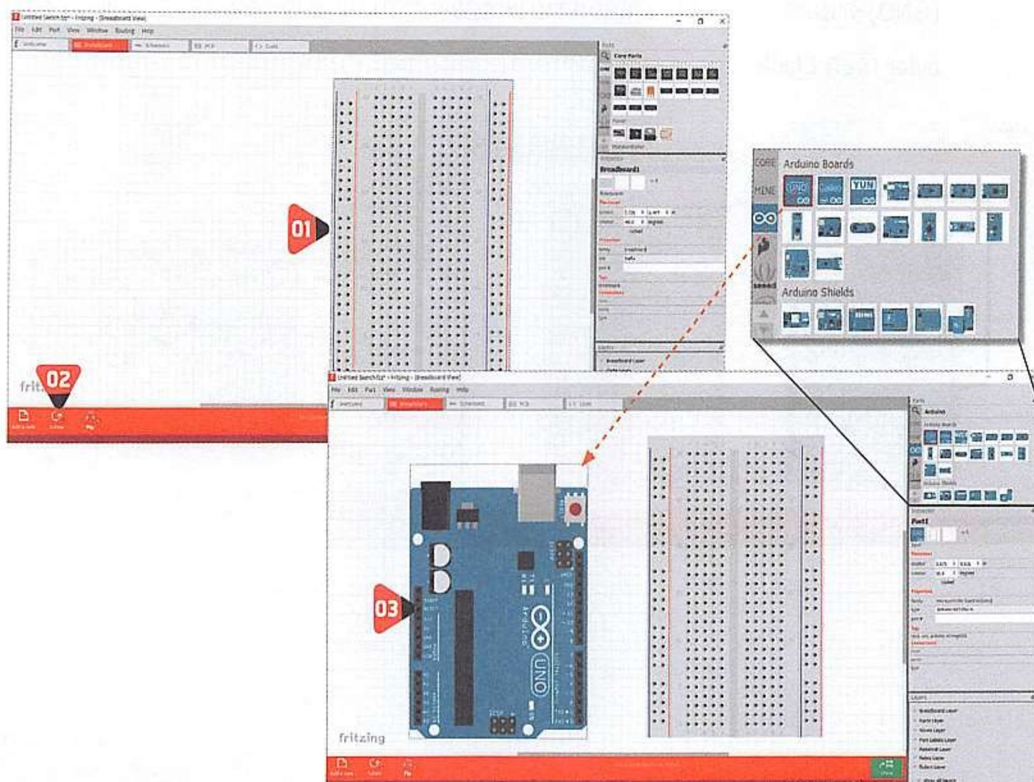
เลื่อนเมาส์ถอยไปข้างหลังเพื่อ Zoom-out

▲ รูปแสดงการ Zoom

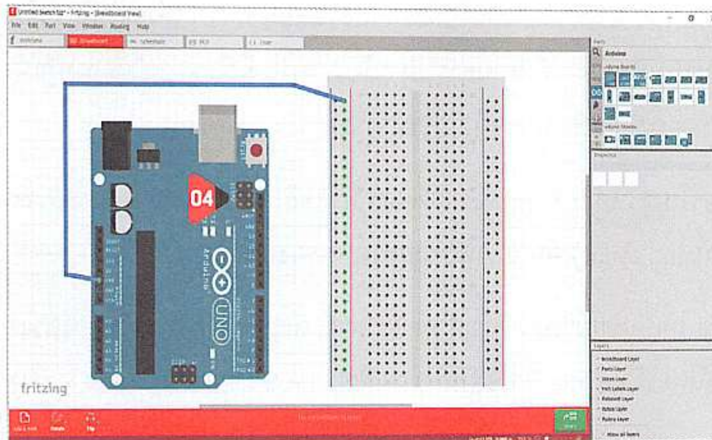
ทดลองเขียนวงจรกับโปรแกรม Fritzing ครั้งแรก

ขั้นตอนต่อไปนี้จะสาธิตการเขียนวงจรที่หน้า Breadboard View เป็นการเขียนวงจรโดยอาศัยชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์ และบอร์ดทดลองเสมือนจริง โดยเราจะนำชิ้นส่วนต่างๆ (Parts) และสายไฟ (Wire) มาวางเชื่อมโยงกันเป็นวงจร ตัวอย่างต่อไปนี้จะแสดงขั้นตอนคร่าวๆ เพื่อความกระชับของเนื้อหา จุดประสงค์ คือ ต้องการเพิ่มพูนทักษะพื้นฐานการใช้งานโปรแกรม เพื่อให้ผู้อ่านได้ลองปฏิบัติตามเท่านั้น ในส่วนนี้จะยังไม่เน้นความเข้าใจเรื่องการทำงานของวงจร

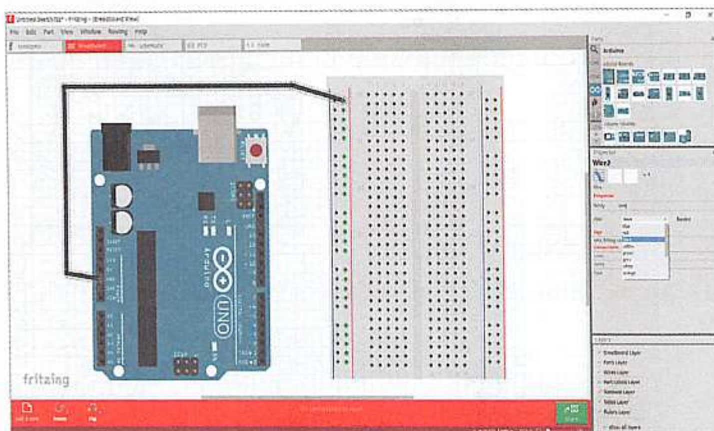
1. ในการเริ่มต้นเขียนวงจรใหม่ ให้คลิกคำสั่ง **New Sketch** หรือคลิกที่แถบ **Breadboard View** จะพบเลย์เอาต์หลัก (Main Layout) คือ มีภาพของบอร์ดทดลองปรากฏขึ้นมา
2. เลือกขนาด **Breadboard** ให้เป็นแบบ **half+** แล้ว**หมุน 90 องศา** ดังรูป (สามารถขยับบอร์ดวางในตำแหน่งที่ต้องการได้โดยใช้วิธี Drag Mouse คือ การคลิกเมาส์ปุ่มซ้าย (ค้าง) > ลาก > วาง)
3. ไปที่ **Parts** คลิกที่หมวด **Arduino** (สัญลักษณ์โลโก้) แล้ว Drag Mouse บอร์ด **Arduino UNO** มาวางข้าง Breadboard แล้วหมุนบอร์ด ดังรูป



4. ต่อสายกราวด์ เริ่มจากคลิกที่ขา **GND** บนบอร์ด Arduino ลากไปยัง **Negative (-)** บน Breadboard และคลิกที่สายไฟเพื่อปรับแต่งการเดินสายให้เหมาะสม ดังรูป (ลากสายและดึงสายด้วยวิธี Drag Mouse เช่นกัน)



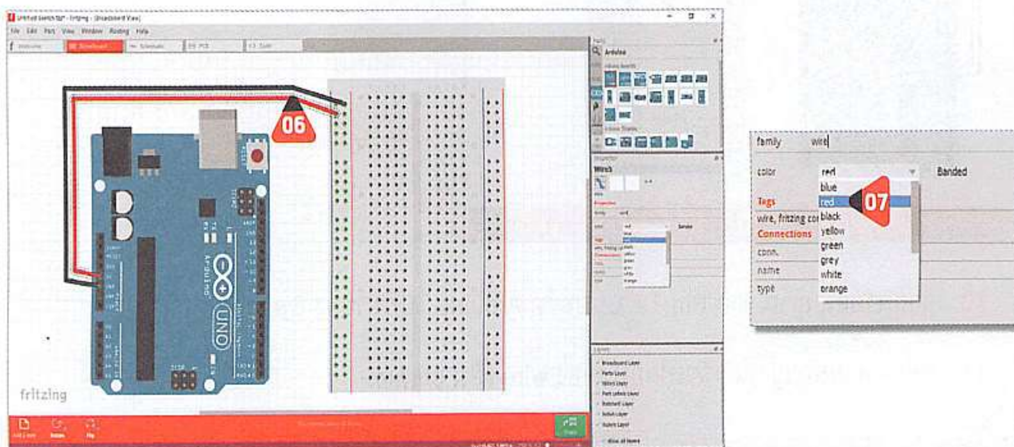
5. ต่อไปให้เปลี่ยนสีสายไฟจากสีน้ำเงิน (หรือสีอื่น ๆ) ให้เป็นสีดำ ซึ่งเป็นสีที่สงวนไว้ใช้กับกราวด์ (GND) ตามหลักสากล ให้คลิกเลือกสายไฟที่ต้องการเปลี่ยนสี แล้วคลิกปุ่ม **Dropdown** ตรง **color** เลือก **black**



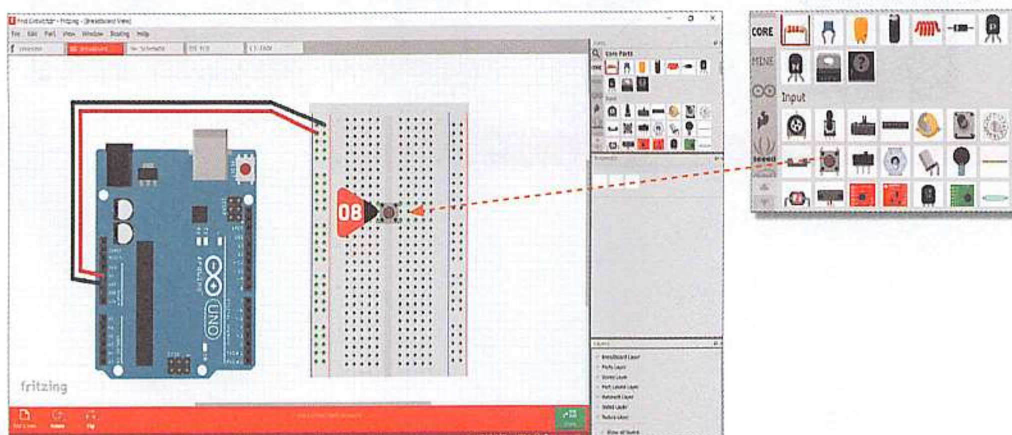


6. ต่อสาย Power เริ่มจากขา **5 V** บนบอร์ด Arduino ลากสายไปยัง **Positive (+)** บน Breadboard และคลิกที่สายไฟเพื่อปรับแต่งการเดินสายให้เหมาะสม ดังรูป

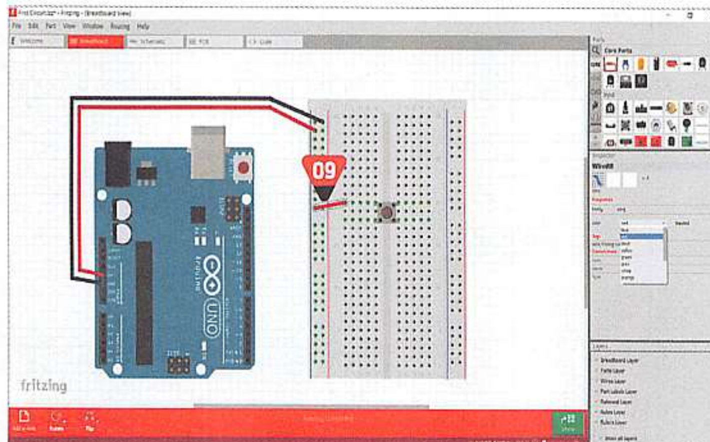
7. เปลี่ยนสีสายไฟเป็นสีแดง (Power) ซึ่งเป็นสีที่สงวนไว้ใช้กับเพาเวอร์ตามหลักสากล คลิกปุ่ม **Dropdown** ตรง **color** เลือก **red**



8. เพิ่ม **Parts** ในหมวด **CORE** คลิกเลือก **Pushbutton** (Switch) แล้วลากมาวางคร่อมตรงเส้นแบ่งครึ่งบน Breadboard หมุนภาพ 90 องศา ขยับให้ขาของ Pushbutton ตรงกับรูของ Breadboard ดังรูป

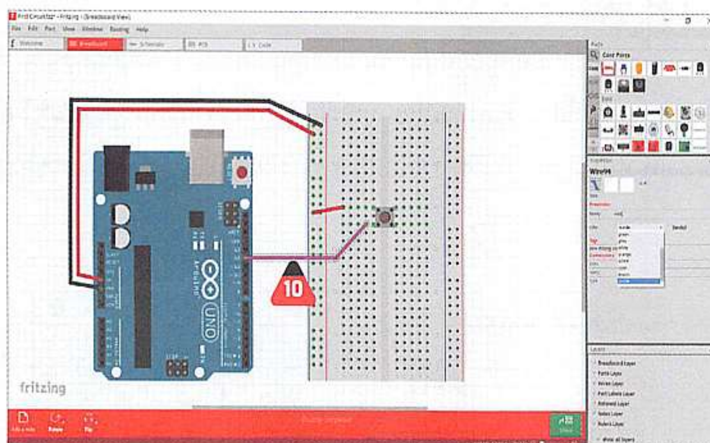


9. บน Breadboard ให้ต่อสายไฟจาก **Positive** ไปที่ขาของ **Switch** (กำหนดสี **wire** เป็น **red**)



10. ลากสายสัญญาณจาก **Pin 12** บนบอร์ด Arduino ไปที่ขาของ **Switch**

11. คลิกที่สายสัญญาณ แล้วเปลี่ยนสีของ **wire** เป็น **purple**

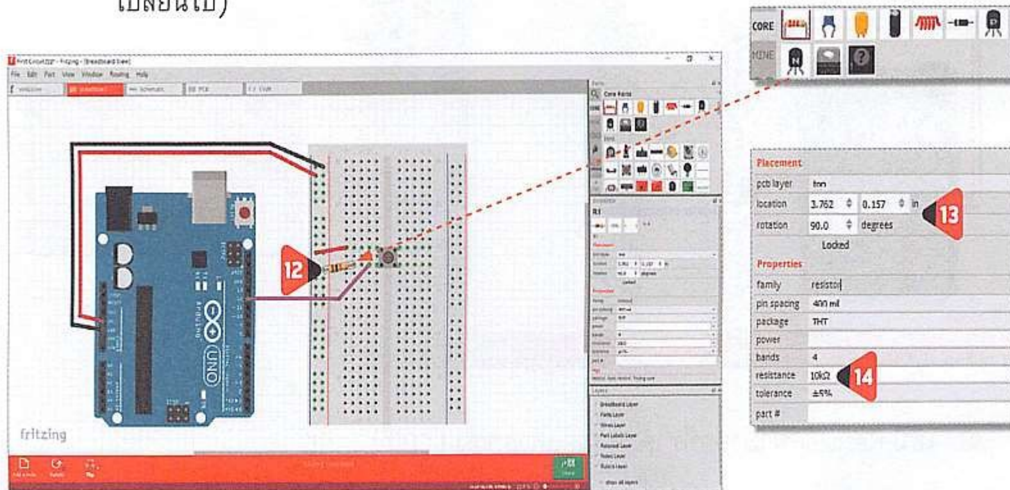




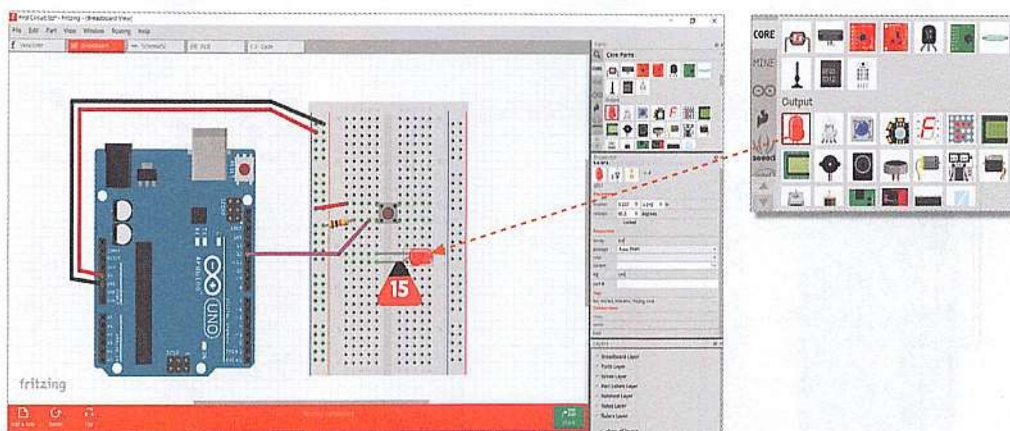
12. เพิ่ม **Resistor** (ตัวต้านทาน) ไปที่ **Parts** ในกลุ่ม **CORE** มองหา **Resistor** แล้วลากมาวางในตำแหน่งดังรูป

13. ปรับตำแหน่ง Resistor แบบละเอียดได้ที่ **location** และ **degrees** และคลิกที่ปลายขา Resistor ให้ขาจัมป์กับรูทั้งสองด้าน สังเกตจะขึ้นจุดสีเหลืองเป็นไกด์ทางเดินวงจร ถ้าไม่ขึ้นแสดงว่าวงจรยังไม่เชื่อมกัน (ดู Note เพิ่มเติม)

14. เปลี่ยนค่า **resistance** ให้เป็นค่าที่ต้องการ ในที่นี้คือ 10k (สังเกตโค้ดสีที่ตัวต้านทานจะเปลี่ยนไป)

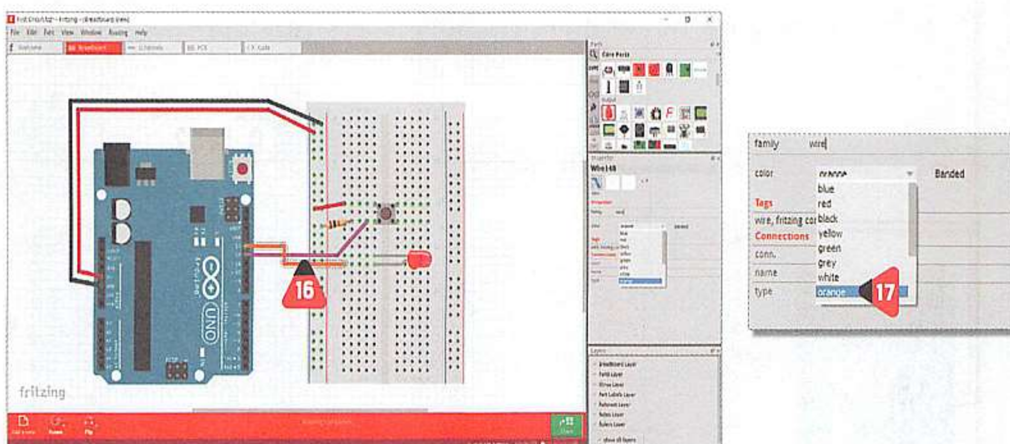


15. เพิ่มหลอด **LED** ซึ่งจะเก็บอยู่ในกลุ่ม **CORE** เช่นกัน คลิกลากมาวางในตำแหน่งดังรูป



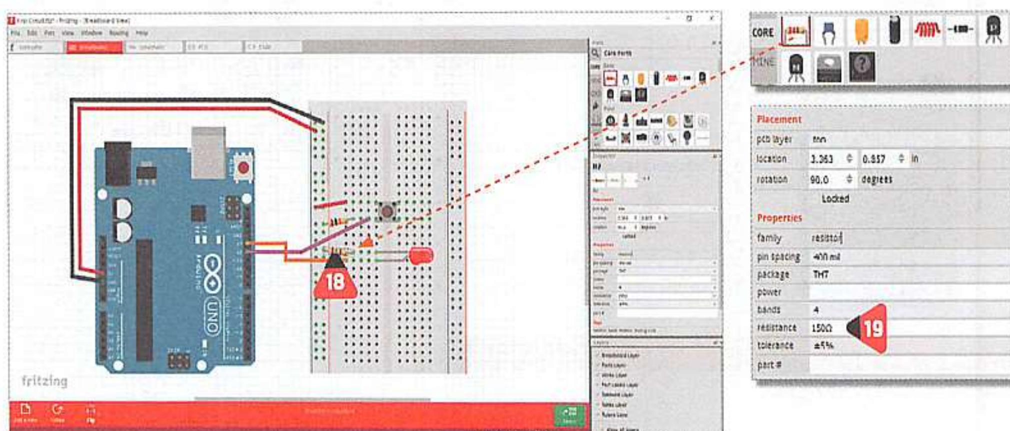
16. ต่อสาย Power เข้าที่หลอด LED (สังเกตขั้วบวกจะอยู่ฝั่งขาที่หักออกเช่นเดียวกับตัวหลอด ส่วนขั้วลบจะเรียบตรง) โดยลากสายจากขา **Pin 13** ของบอร์ด Arduino ไปยัง**ขั้วบวก (+)** ของหลอด LED

17. หลังจัดสายเสร็จเรียบร้อยแล้ว ให้เปลี่ยนสีสายไฟและหลอด **LED** เป็น**สีส้ม** เพื่อสื่อความหมายหรือความสัมพันธ์ระหว่างสายไฟและหลอด LED



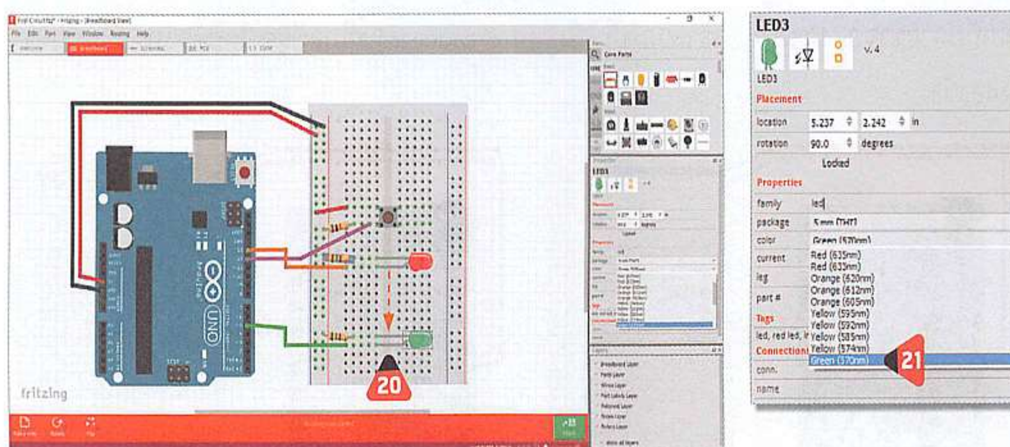
18. เพิ่ม Resistor นำมาต่อที่ขาด้าน **Negative** ของ LED กับ **GND**

19. เปลี่ยนค่าเป็น **150 โอห์ม** (Ohms) อย่าลืมคลิกที่ขาของ Resistor ทั้งสองข้างเพื่อให้มั่นใจว่ามันเชื่อมต่อกับวงจรเรียบร้อยแล้วแล้ว (สังเกตจะขึ้นจุดไกด์สีเหลืองเพื่อแสดงว่า มันเชื่อมโยงกับ Parts อะไรบ้าง)



20. เพิ่มวงจร LED ชุดที่ 2 เราสามารถ Copy ได้จากชุดแรก โดยกดปุ่ม <Ctrl + ออบเจ็กต์ที่ต้องการ> เมื่อเลือกครบแล้วให้กดปุ่ม <Ctrl + C> เพื่อ Copy และ <Ctrl + V> เพื่อ Paste แล้วขยับชุด LED วางในตำแหน่งตามรูป

21. เปลี่ยนสีสายไฟและสีหลอด LED เป็น Green

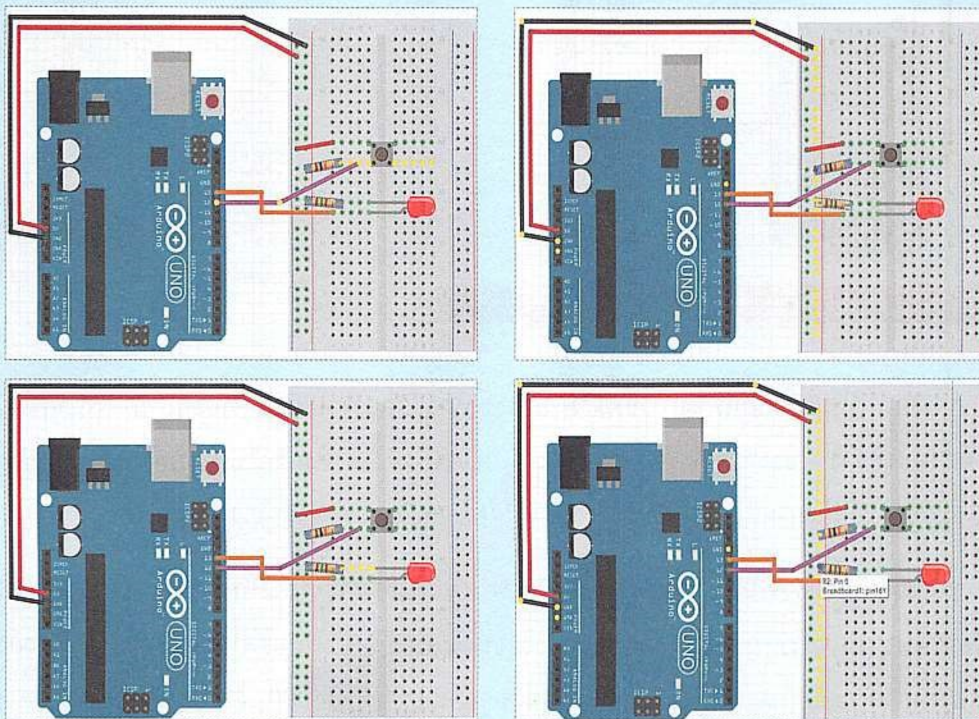


ขั้นตอนทั้งหมดนี้ เป็นเพียงการแสดงตัวอย่างของการต่อวงจรโดยใช้โปรแกรมในครั้งแรกเท่านั้น ยังไม่ต้องสนใจว่า คือวงจรอะไร ทำงานอย่างไร วงจรใช้งานได้จริงหรือไม่ แค่อยากให้เห็นภาพว่า การเขียนวงจรนั้นไม่ยากเลย ใช้ทักษะธรรมดาๆ โดยการคลิก-ลาก-วาง-กำหนดค่า

และเมื่อเขียนวงจรในหน้า Breadboard View สำเร็จแล้ว ให้ลองคลิกที่ Schematic และ PCB View จะพบว่า มันจะสร้างแผนผังวงจร และลายวงจรบนแผ่น PCB ให้โดยอัตโนมัติ (แต่เส้นสายจะยังดูสับสน จำเป็นต้องปรับแต่งให้ดูเรียบร้อยขึ้น) หลังจากนี้แนะนำให้หาวงจรในอินเทอร์เน็ตมาลองทำตาม เพื่อพัฒนาทักษะเพิ่มเติม เราสามารถเพิ่มพูนความรู้ต่อยอดได้ด้วยตนเอง

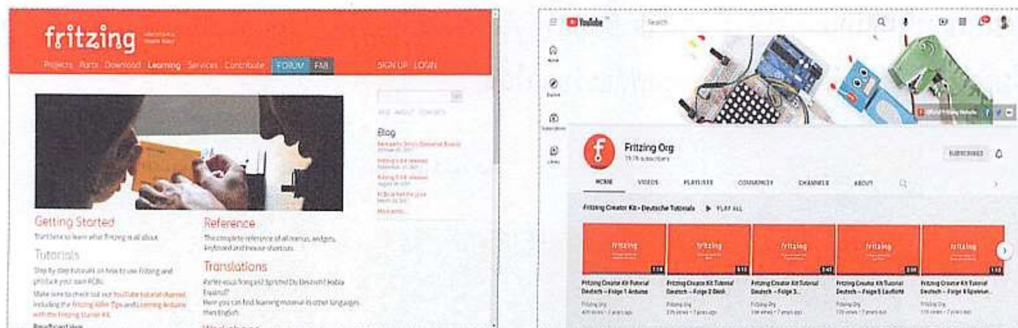
การทดสอบการเชื่อมต่อระหว่างอุปกรณ์กับวงจร

จากตัวอย่างข้างต้น เมื่อเราเพิ่ม Resistor มาวางบน Breadboard แล้ว จำเป็นต้องตรวจสอบให้แน่ใจว่า ขาของ Resistor ได้เชื่อมโยงเข้ากับวงจรส่วนอื่นๆ เรียบร้อยแล้ว วิธีคือ คลิกที่ขา Resistor สังเกตว่าจะมีจุด หรือ Dots ไกด์สีเหลืองปรากฏหรือไม่ ถ้าไม่ขึ้นจุดไกด์สีเหลือง ให้แก้ไขด้วยการคลิกที่ปลายขาของ Resistor แล้วลากไปให้ถึงรูบน Breadboard ที่ใกล้ที่สุด โดยต้องทดสอบกับขาทั้งสองข้าง



แหล่งศึกษาการใช้งานโปรแกรม Fritzing เพิ่มเติมทางออนไลน์

เนื่องจากหนังสือเล่มนี้เน้นไปที่กระบวนการการเรียนรู้ และสร้างทักษะผ่านการทำ Lab และ Project ไม่ได้เน้นสอนพื้นฐานการใช้งานโปรแกรม หรือการเขียนโปรแกรม ผู้อ่านจึงจำเป็นต้องศึกษาด้วยตนเอง โดยสามารถเข้าไปเรียนรู้ได้ที่เว็บไซต์ของ Fritzing ที่เมนูหลัก Learning ในหัวข้อ Getting Started ซึ่งจะมีบทเรียนให้ศึกษาพื้นฐานครบทุกเรื่อง อีกทั้งยังมี YouTube Tutorial Channel ให้ดูในรูปแบบวิดีโออีกด้วย

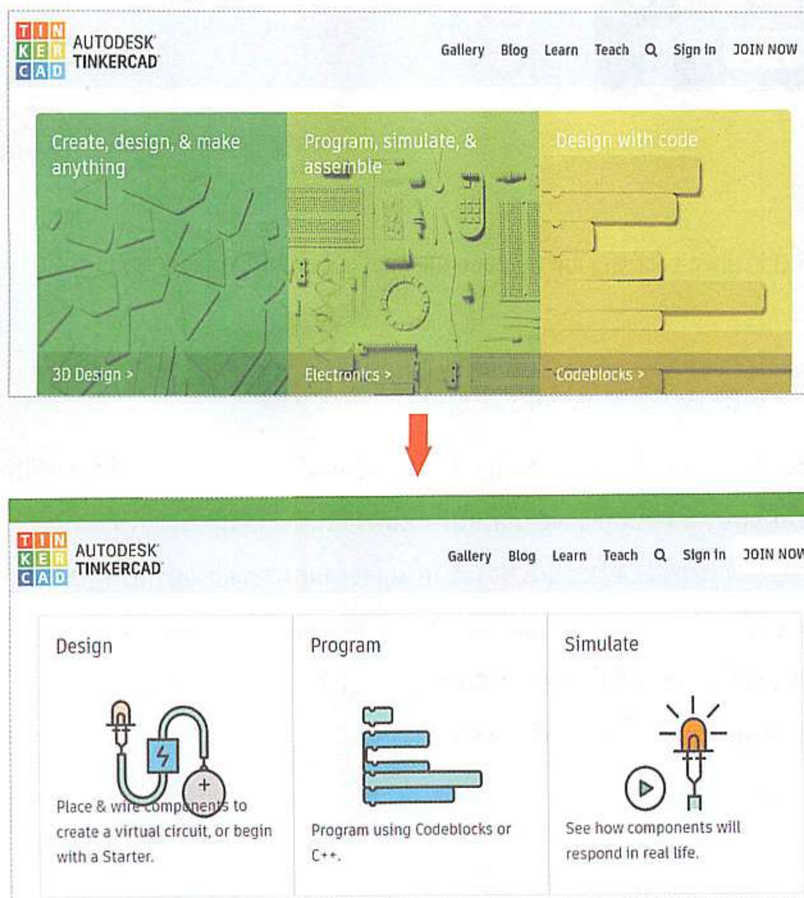


▲ รูปแสดงหน้าเว็บเพจ Learning และ YouTube Channel ของเว็บไซต์ Fritzing สำหรับศึกษาการใช้งาน

Tinkercad (Arduino Simulator)

Tinkercad คือ คอลเล็คชันของซอฟต์แวร์ในแบบฟรีออนไลน์ เป็นเครื่องมือที่จะช่วยในการฝึกคิดก่อนที่จะสร้างมันขึ้นมาจริงๆ แต่ที่เราจะเน้นกันในบทนี้คงมุ่งไปที่เครื่องมือที่ช่วยในการพัฒนา Program, Simulate, Assembly ในทาง Electronics ซึ่งจะช่วยในการจำลองการต่อวงจรเพื่อทดสอบก่อนต่อวงจรเพื่อใช้งานจริง ช่วยในการออกแบบและลดความเสียหาย หากวงจรที่เราออกแบบนั้นผิดพลาด นอกจากนี้ยังช่วยในการพัฒนาโปรแกรมทั้งในรูปของ Code (ภาษาโปรแกรมแบบข้อความ) หรือ CodeBlocks (ภาษาโปรแกรมแบบกราฟิกที่ใช้การคลิก-ลาก-วาง)

Tinkercad มีเครื่องมือที่จะช่วยให้เราสามารถต่อวงจรอิเล็กทรอนิกส์ โดยใช้อุปกรณ์และส่วนประกอบที่เป็นภาพกราฟิกเสมือนจริง มีพื้นที่สำหรับเขียนโปรแกรม และประมวลผลเพื่อทดสอบการทำงาน และดูว่าได้ผลลัพธ์เป็นไปตามวัตถุประสงค์หรือไม่ สามารถแก้ไขโดยการเพิ่ม-ลด หรือเปลี่ยนชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์ได้ ช่วยลดค่าใช้จ่ายในช่วงออกแบบได้มาก เพราะไม่ต้องจัดหาอุปกรณ์มากมายมาไว้ทดสอบ โดยเฉพาะมือสมัครเล่น เช่น อยากรู้ว่าบอร์ด Arduino รุ่นไหนเหมาะกับขนาดโปรเจ็กต์ที่เรากำลังทำอยู่ โดยสามารถเลือกบอร์ดจำลองรุ่นต่างๆ มาลองทดสอบดูได้ เพราะมันสามารถแสดงผลการทำงานของ Arduino แบบออนไลน์ได้ โดยไม่จำเป็นต้องใช้งานบอร์ดจริง ช่วยให้เราเปรียบเทียบการทำงานก่อนสั่งซื้อบอร์ดจริงได้ ข้อเสียเท่าที่ลองใช้งาน คือ อาจจะมีดีเลย์บ้างในบางจังหวะ ซึ่งก็เป็นธรรมดาของซอฟต์แวร์ออนไลน์

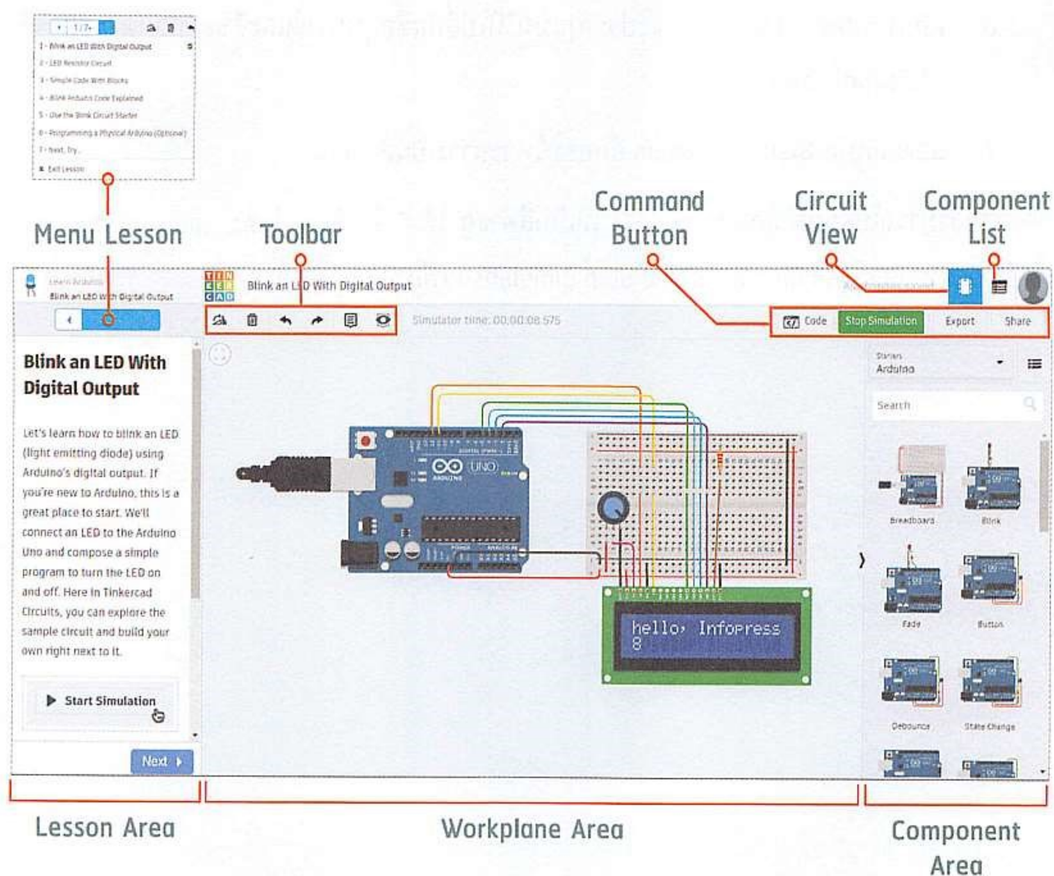


▲ รูปแสดงเว็บเพจ www.tinkercad.com ในหมวด Electronics สำหรับ Design, Program และ Simulate

ส่วนประกอบของเว็บเพจ Learn Arduino

หน้าเว็บเพจ Learn Arduino ที่เราจะใช้ในการเรียนรู้ Arduino จะแบ่งออกเป็น 3 ส่วนหลักๆ ได้แก่ Lesson Area (ส่วนบทเรียน), Workplane Area (ส่วนพื้นที่สำหรับต่อวงจร/ประกอบชิ้นส่วน) และ Component Area (ส่วนแสดงรายการอุปกรณ์) ส่วนด้านบนของแต่ละพื้นที่จะเป็นแถบเครื่องมือ (Toolbar)

- **เครื่องมือของ Lesson Area** จะเป็น Menu Lesson มีปุ่มคลิกเดินหน้า-ถอยหลัง หรือ Move ไปยังหัวข้อที่ต้องการได้
- **เครื่องมือของ Workplane Area** จะมีไอคอนพื้นฐาน ได้แก่ Rotate, Trash, Undo, Redo, Annotation และ View/Hide
- **ปุ่มคำสั่งใช้งานทั่วไป** ได้แก่ Code, Start/Stop Simulation, Export และ Share (ปุ่มส่วนนี้ไม่ได้เกี่ยวข้องโดยตรงกับ Component Area แต่เป็นปุ่มใช้งานหลักทั่วไป)



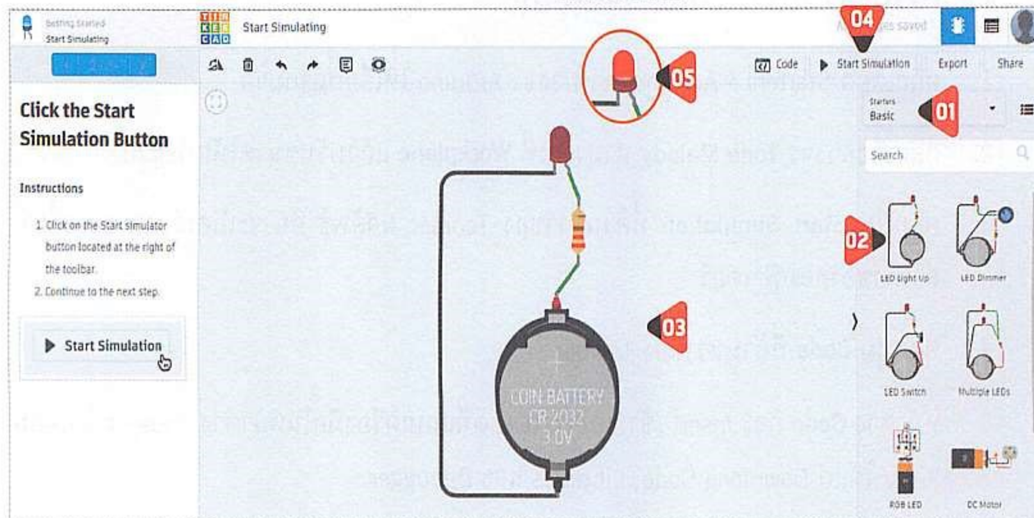
ทดลอง LED Light Up

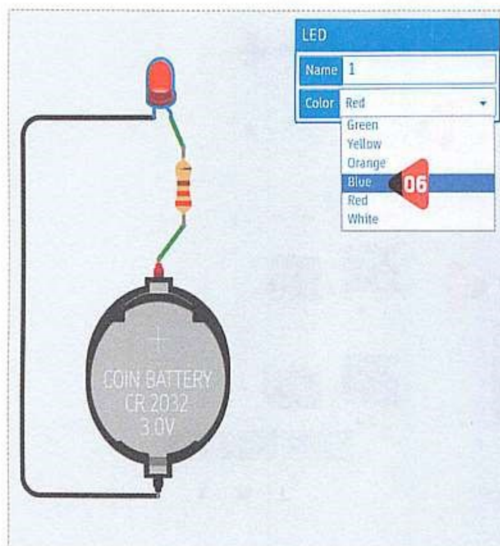
เป็นตัวอย่างวงจรง่ายๆ ที่เหมาะจะนำมาใช้เพื่อศึกษาการใช้งาน Tinkercad ให้เข้าใจเสียก่อน โดยตัวอย่างแรกนี้จะยังไม่มีการเขียนโค้ด

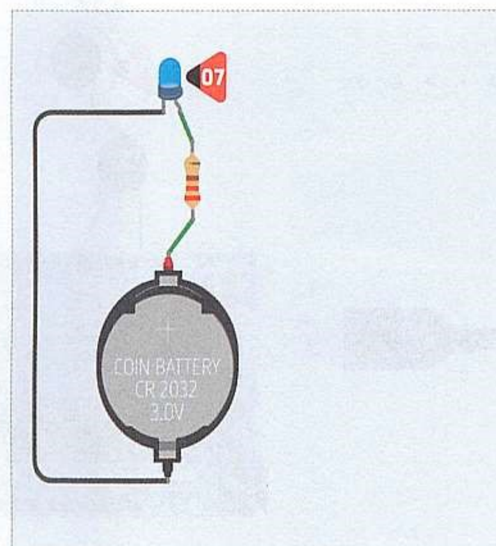
1. คลิกเลือก **Starters > Basic**
2. คลิกเลือกวงจร **LED Light Up**
3. ปรับขนาดภาพให้เหมาะสม โดยวางเมาส์ตรง Workplane (พื้นที่สีเทา) แล้วหมุนล้อเมาส์ (Wheel) ไปข้างหน้าเพื่อขยายใหญ่ขึ้น หรือหมุนถอยหลังเพื่อย่อภาพให้เล็กลง
4. สังเกตด้านซ้ายเป็นส่วนแสดงคำแนะนำว่า ต้องทำอะไรแบบทีละ Step ในที่นี้ให้คลิกปุ่ม **Start Simulation** ที่ด้านขวาของ Toolbar
5. ผลลัพธ์ คือ หลอด LED สีแดงสว่าง
6. คลิกที่หลอด LED สีแดง จะปรากฏเมนูให้เปลี่ยนชื่ออุปกรณ์และสีของหลอด LED ได้ ให้เลือกสีน้ำเงิน (Blue)
7. ลองคลิกปุ่ม **Start Simulation** อีกครั้ง ไฟจะสว่างเป็นสีน้ำเงิน

เราสามารถเปลี่ยนสีให้กับหลอด LED ให้เป็นสีต่างๆ ได้ 6 สี ได้แก่ สีแดง, เขียว, เหลือง, ส้ม, น้ำเงิน และขาว เสร็จแล้วอย่าลืมคลิกปุ่ม Stop Simulation เพื่อหยุดการจำลอง

หลอดสว่าง

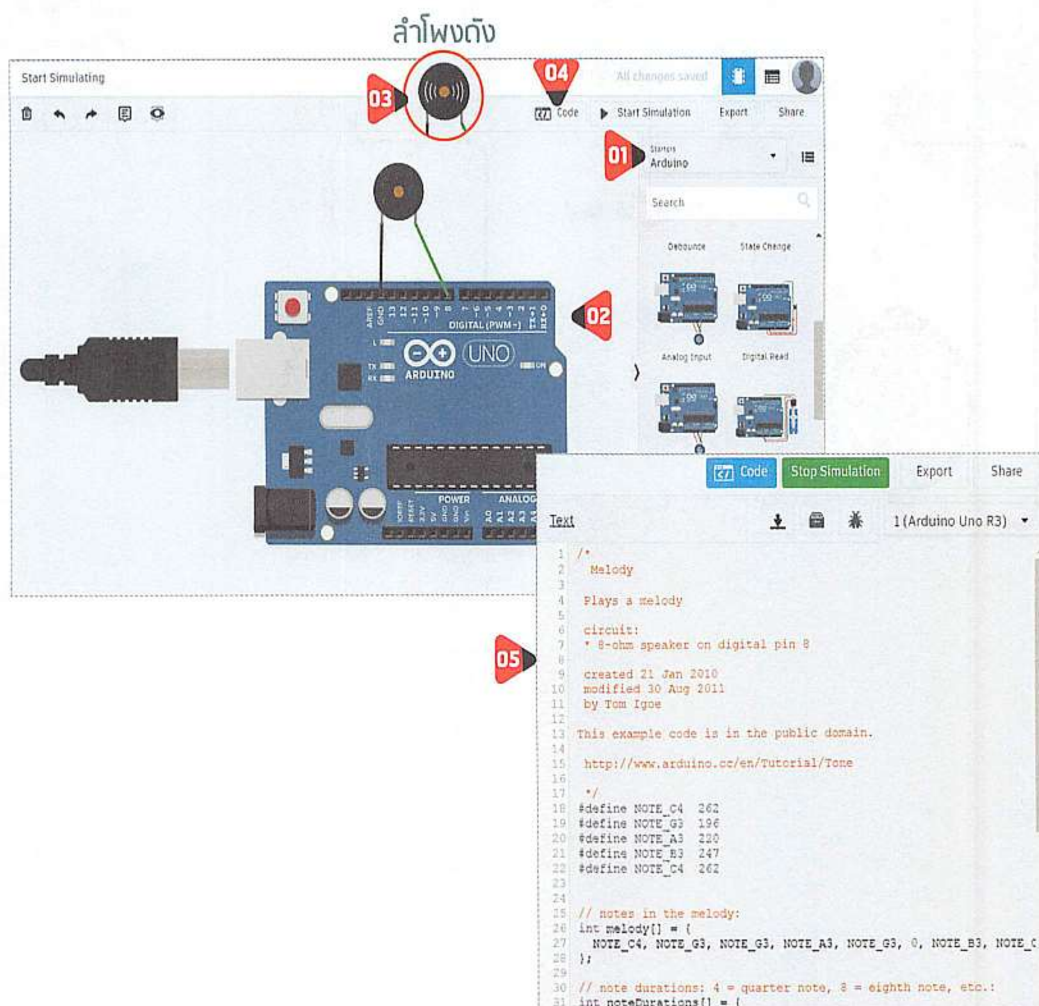

Start Simulating
Code
Start Simulation
Export
Share
Click the Start Simulation Button
Instructions
1. Click on the Start simulator button located at the right of the toolbar.
2. Continue to the next step.
Start Simulation
Basic
Search
LED Light Up
LED Dimmer
LED Switch
Multiple LEDs
RGB LED
DC Motor
COIN BATTERY CR 2032 3.0V
01
02
03
04
05


LED
Name
1
Color
Red
Green
Yellow
Orange
Blue
Red
White
06
COIN BATTERY CR 2032 3.0V


07
COIN BATTERY CR 2032 3.0V

ทดลองต่อวงจร Tone Melody

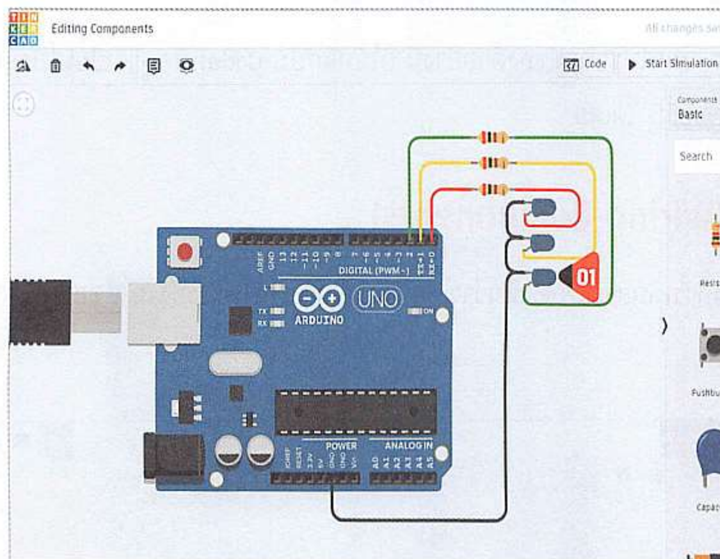
1. คลิกเลือก **Starters > Arduino** จะพบวงจร Arduino ให้เลือกมากมาย
2. คลิกเลือกวงจร **Tone Melody** นำมาวางที่ **Workplane** แล้วปรับขนาดให้เหมาะสม
3. คลิกปุ่ม **Start Simulation** ที่ด้านขวาของ Toolbar ผลลัพธ์ คือ จะเกิดเสียงเพลงดังขึ้นที่ลำโพงของคอมพิวเตอร์
4. คลิกปุ่ม **Code** ที่ด้านขวาของ Toolbar
5. หน้าต่าง **Code** ก็จะ Insert เข้ามาทันที สังเกตที่แถบเครื่องมือในหน้าต่าง Code จะมีไอคอน 3 ปุ่ม ได้แก่ Download Code, Libraries และ Debugger

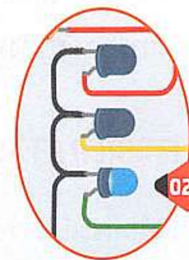


การแก้ไขส่วนประกอบ (Editing Components)

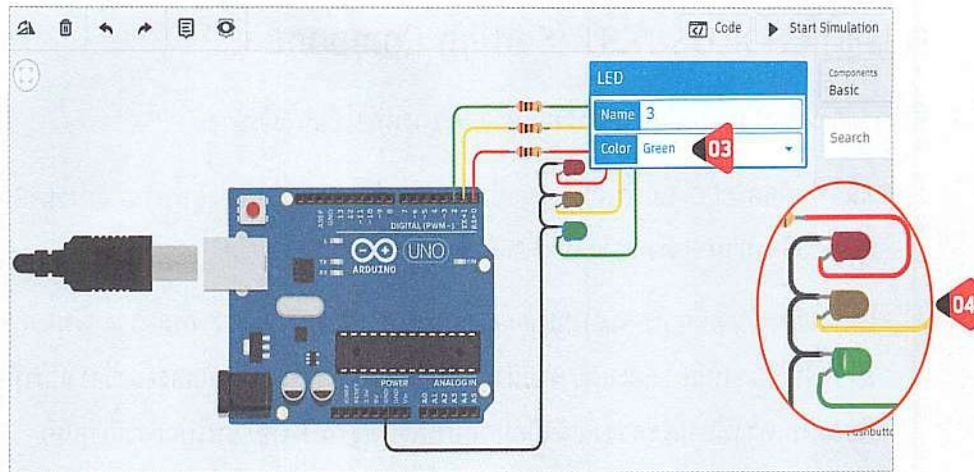
ทีนี้เราลองนำวงจรมาเปลี่ยนอุปกรณ์กันดูว่า สามารถทำได้อย่างไรบ้าง

1. คลิกที่หลอด LED บนพื้นที่ทำงาน ซึ่งประกอบด้วยหลอด LED สีน้ำเงิน จำนวน 3 ดวง, Resistor จำนวน 3 ตัว และบอร์ด Arduino
2. ในตอนนี้ถ้าเราคลิกปุ่ม **Start Simulation** หลอดไฟสีน้ำเงินจะสว่างทีละดวง เริ่มจากหลอดสายไฟสีเขียว เหลือง และแดง ตามลำดับ เมื่อไฟดวงแรกดับ ดวงที่สองจะสว่าง และเมื่อดวงที่สองดับ ดวงที่สามจะสว่าง แล้วย้อนกลับไปเริ่มดวงที่ 1 ใหม่ วนไปอย่างนี้เรื่อยๆ





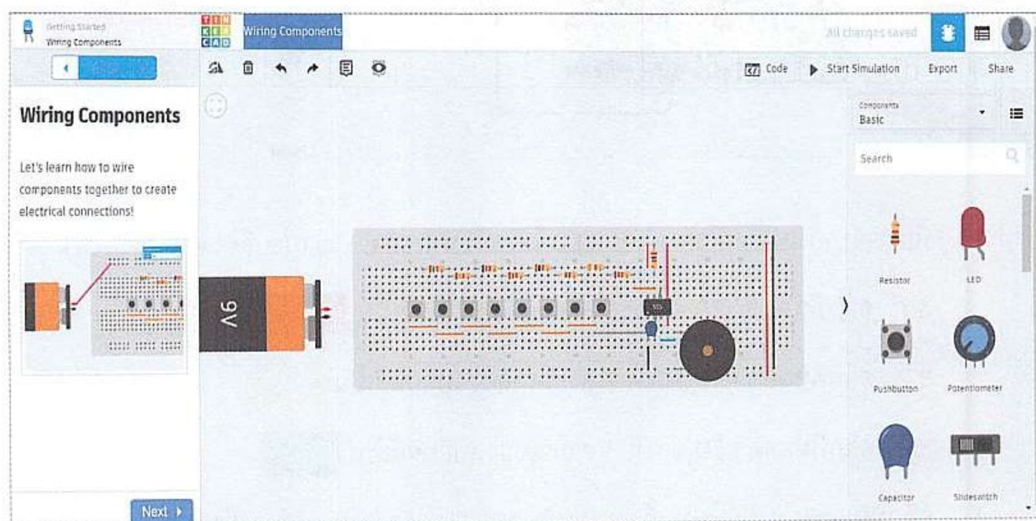
3. ทีนี้เราจะลองเปลี่ยนสีของหลอด LED ให้เหมือน Traffic Light (เปลี่ยนตามสีสายไฟ)
   - 3.1 คลิกที่หลอด LED ดวงที่ 1 สายสีแดง เปลี่ยนสีเป็น Red
   - 3.2 คลิกที่หลอด LED ดวงที่ 2 สายสีเหลือง เปลี่ยนสีเป็น Yellow
   - 3.3 คลิกที่หลอด LED ดวงที่ 3 สายสีเขียว เปลี่ยนสีเป็น Green
4. คลิกปุ่ม **Start Simulation** ก็จะพบว่าหลอด LED จะสว่างตามสีที่เรากำหนด



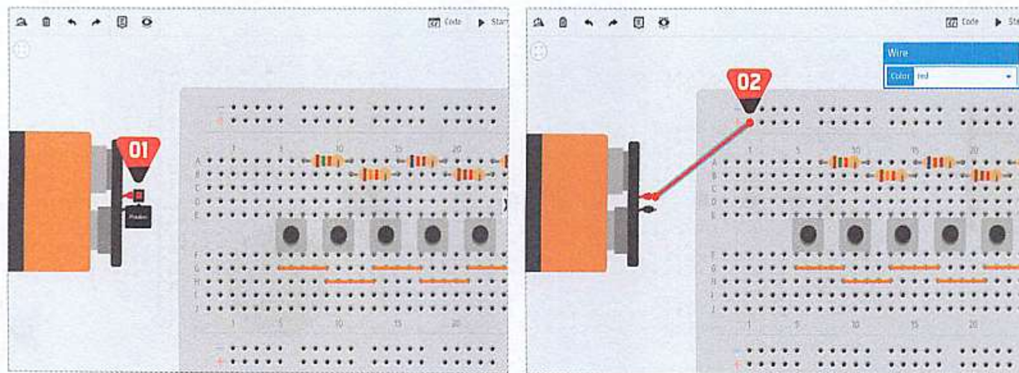
หากต้องการศึกษาการเขียนโปรแกรมของตัวอย่างนี้ ก็ให้คลิกปุ่ม **Code** เพื่อดูวิธีเขียนโปรแกรมควบคุมการติดดับของหลอด LED ได้เลย

การเดินสาย (Wiring Components)

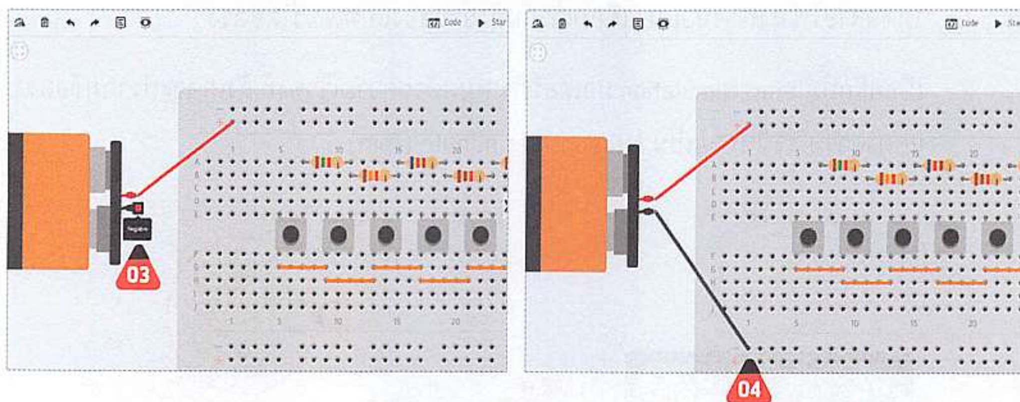
ต่อไปนี้จะเป็นตัวอย่างการเดินสายไฟ จะเห็นว่าวงจรนี้ยังไม่สมบูรณ์ เพราะยังไม่ได้ต่อสายไฟจากแบตเตอรี่ไปยัง Breadboard



1. เดินสายไฟจากขั้วบวก (+) สีแดงของแบตเตอรี่ (ขนาด 9 V) เพื่อเชื่อมไปยังวงจรที่เราต่อไว้แล้วบนบอร์ดทดลอง ให้นำเมาส์ไปคลิกที่**ขั้วบวก (สีแดง) ของแบตเตอรี่** (สังเกตคำว่า 'Positive' จะปรากฏขึ้นมา)
2. ลากสายไปคลิกตรงสัญลักษณ์ + **ที่บอร์ดทดลอง** เท่านี้ก็ Wiring เสร็จแล้ว (อย่าลืมเปลี่ยนสีสายไฟเป็นสีแดง)

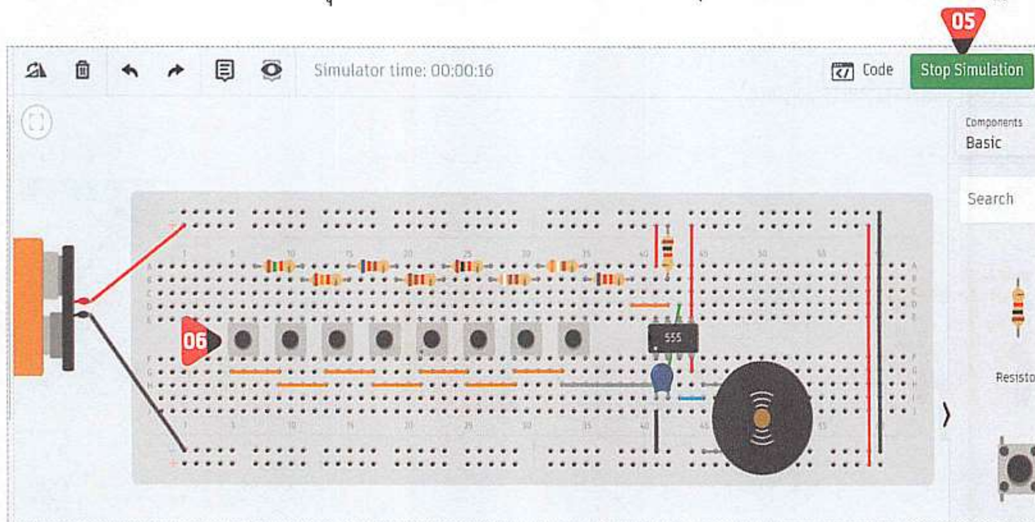


3. คลิกเมาส์ที่**ขั้วลบ (-) สีดำ** ของแบตเตอรี่ (สังเกตคำว่า 'Negative' จะปรากฏขึ้นมา)
4. ลากสายไปคลิกตรงสัญลักษณ์ - **ที่บอร์ดทดลอง** เท่านี้ก็ Wiring เสร็จแล้ว (อย่าลืมเปลี่ยนสีสายไฟเป็นสีดำ)



5. เมื่อเดินสายไฟเสร็จแล้ว ให้คลิกปุ่ม **Start Simulation**

6. สังเกตว่าบน Breadboard จะมีสวิตช์ 8 ปุ่ม ขณะที่สถานะ Simulator กำลังทำงาน หากลองนำเมาส์ไปคลิกทีละปุ่มก็จะได้ยินโทนเสียงที่แตกต่างกัน (คลิกค้างก็จะได้ยินเสียงยาวๆ)

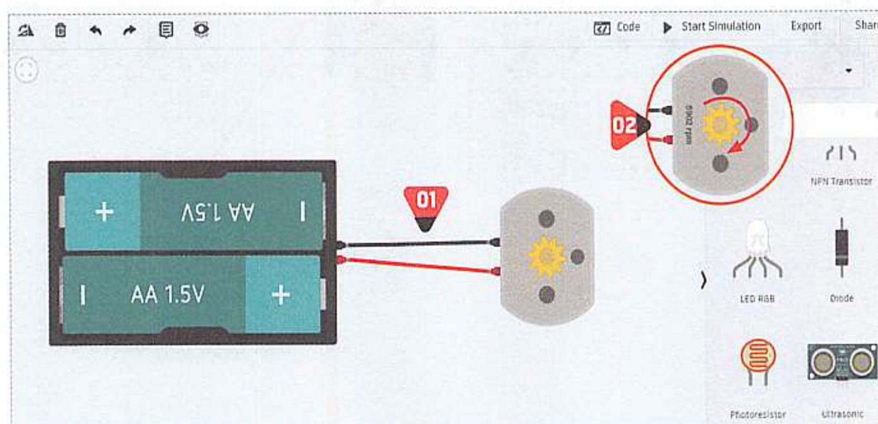


การเพิ่มชิ้นส่วนอุปกรณ์เพิ่มเติม (Adding Components)

สำหรับตัวอย่างนี้จะเป็นการเรียนรู้ว่า เราจะเพิ่มชิ้นส่วนเพื่อควบคุมความเร็วมอเตอร์ได้อย่างไร

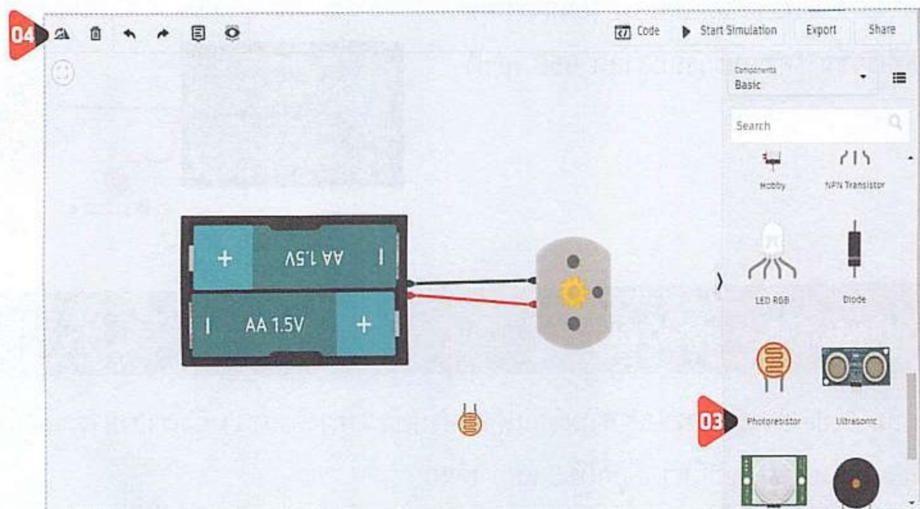
1. ให้ต่อสายไฟระหว่างมอเตอร์เชื่อมเข้ากับลังถ่านของแบตเตอรี่โดยตรง

2. เมื่อคลิกปุ่ม **Start Simulation** มอเตอร์เริ่มหมุนด้วยความเร็วคงที่ สังเกตจะปรากฏข้อความแสดงความเร็วมีหน่วยเป็น rotations per minute (rpm)



3. คลิกปุ่ม **Stop Simulation** แล้วหา Component ที่ชื่อ **Photoresistor** มาวางในพื้นที่งาน ซึ่งอุปกรณ์ตัวนี้ใช้ในการปรับความเร็วรอบมอเตอร์

4. คลิกที่ตัว **Photoresistor** แล้วคลิกที่ไอคอน **Rotate** เพื่อหมุนภาพให้ขาชี้ขึ้นข้างบน (หรือกดปุ่ม <R> ซ้ำๆ บนคีย์บอร์ด)



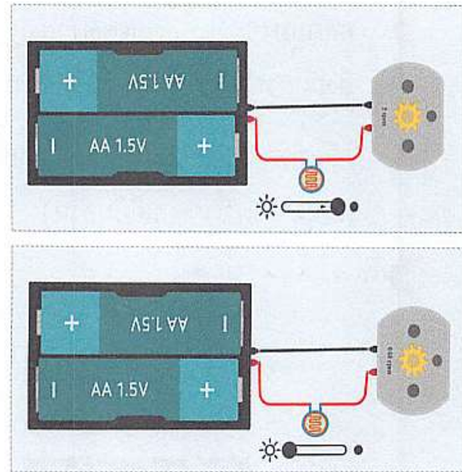
5. ต่อไปให้ลบสายไฟสีแดงออก โดยคลิกเลือกสายสีแดงแล้วคลิกที่ไอคอน **Trash** (กดปุ่ม <Delete> ก็ได้)

6. ลากสายจากขา **Photoresistor** ไปต่อเข้ากับขั้วบวก **Positive** ของแบตเตอรี่ ส่วนขาอีกข้างต่อเข้ากับ **Terminal 2** ของมอเตอร์

7. คลิกปุ่ม **Start Simulation** เพื่อทดสอบการทำงาน



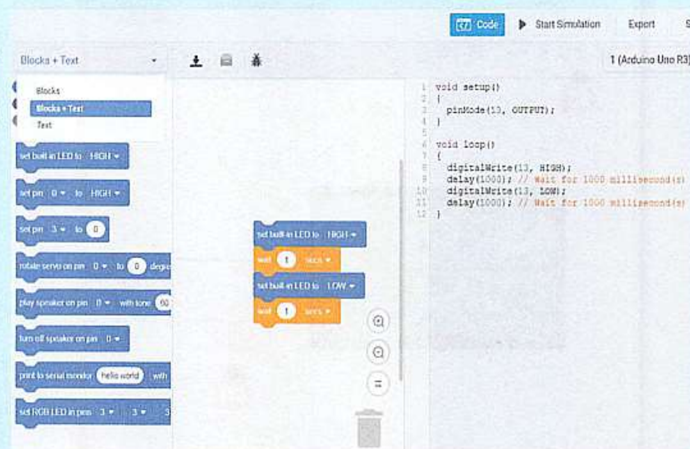
จะพบแถบ Slider ของ Potentiometer ปรากฏขึ้นมา (ถ้าไม่พบให้เอาเมาส์คลิกที่ตัว Photoresistor) จะเห็นว่ามอเตอร์จะหมุนด้วยความเร็วต่ำ (ความเร็วรอบต่ำสุด 2 rpm) ให้ลองนำเมาส์ไปปรับความเร็ว โดยเลื่อนปุ่มสีดำมาทางซ้าย จะเห็นว่ามอเตอร์หมุนด้วยสปีดที่เร็วขึ้น (ความเร็วรอบสูงสุด 658 rpm)



การสั่งให้แสดง Code ทำอย่างไร

เมื่อคลิกปุ่ม Code ก็จะมีหน้าต่างสำหรับเขียนโค้ดแทรกเข้ามาทางด้านขวา ซึ่งเราสามารถเลือกได้ว่าจะให้แสดงแบบไหน โดยมีทางเลือกให้ 3 แบบ ได้แก่

- Blocks : มีหน้าจอเป็นกราฟิกและเราสามารถลาก CodeBlocks มาวางต่อๆ กัน เพื่อควบคุมการทำงาน วิธีนี้แม้ไม่รู้โครงสร้างภาษาก็สามารถเขียนโปรแกรมได้ เหมาะสำหรับเด็กที่เริ่มต้นหัดเขียนโปรแกรมใหม่ๆ
- Text : การเขียนโค้ดด้วยภาษา C, C++ ที่ต้องเรียนรู้โครงสร้างไวยากรณ์ภาษา ซึ่งมีความซับซ้อน ต้องใช้เวลาพอสมควรในการศึกษา จึงจะสามารถเขียนชุดคำสั่งเพื่อโปรแกรมการทำงานให้กับบอร์ดได้อย่างถูกต้อง
- Blocks + Text : เป็นการแสดงโค้ดทั้งสองแบบที่กล่าวมาข้างต้น



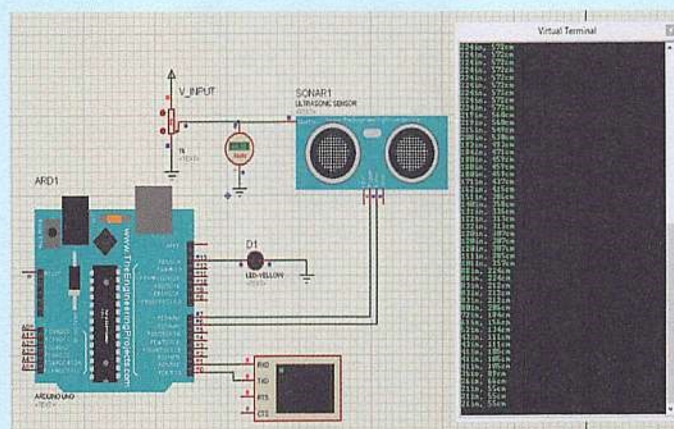


Proteus (Circuit Simulator for Professional)

Proteus คำจำกัดความสั้นๆ คือ "Design Suite"
หรือ "PCB Design and Circuit Simulator Software"

เป็นชุดซอฟต์แวร์ลิขสิทธิ์สำหรับการออกแบบทางอิเล็กทรอนิกส์ ที่มีระบบการทำงานอัตโนมัติ เพื่อสร้างแผนผังวงจร (Schematics) และงานพิมพ์อิเล็กทรอนิกส์ (Electronic Prints) สำหรับโรงงานผลิตแผ่นวงจรพิมพ์ (Print Circuit Board : PCB) โดยจุดเด่นของ Proteus คือ มันเป็นโปรแกรมที่เราสามารถเพิ่ม Library ของ Arduino และ Sensors รุ่นต่างๆ ที่ยังไม่มีใน Library ของ Proteus ได้ กล่าวได้ว่า เป็นซอฟต์แวร์ที่มีการทำงานที่เป็นมืออาชีพมากขึ้น เหมาะสำหรับวิศวกรและช่างเทคนิคในสายงานอิเล็กทรอนิกส์ หรือผู้ที่ต้องการพัฒนาตัวเองให้ก้าวไปสู่ระดับมืออาชีพอย่างแท้จริง

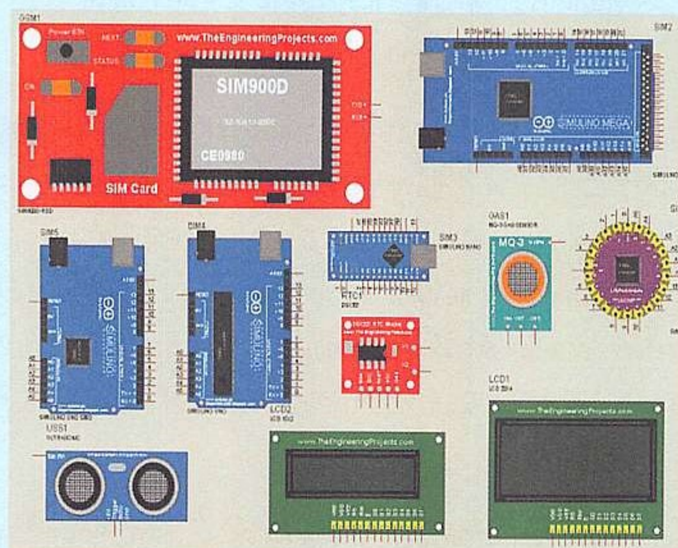
ซอฟต์แวร์ Proteus Professional ไม่ใช่ฟรีแวร์ แต่สามารถดาวน์โหลดมาทดลองใช้งานก่อนได้ โดยดาวน์โหลดตัว Try for Free แม้จะเป็นตัวทดลองแต่ก็ไม่จำกัดเวลาการใช้งาน และใส่ Sample งานออกแบบครอบคลุมการทำงานให้ศึกษาได้อย่างเต็มที่ แต่จะถูกจำกัดในหลายๆ เรื่อง เช่น ไม่อนุญาตให้พรินต์วงจร Schematic และ Layout ของตัวอย่างงานออกแบบ ไม่อนุญาตให้ Save งาน และไม่สามารถ Simulate งานออกแบบไมโครคอนโทรลเลอร์ของเราได้ โดยสามารถสั่งซื้อและดาวน์โหลดโปรแกรม Proteus ได้ที่เว็บไซต์ www.labcenter.com



▲ รูปแสดงหน้าจอการใช้งานภายในโปรแกรม Proteus



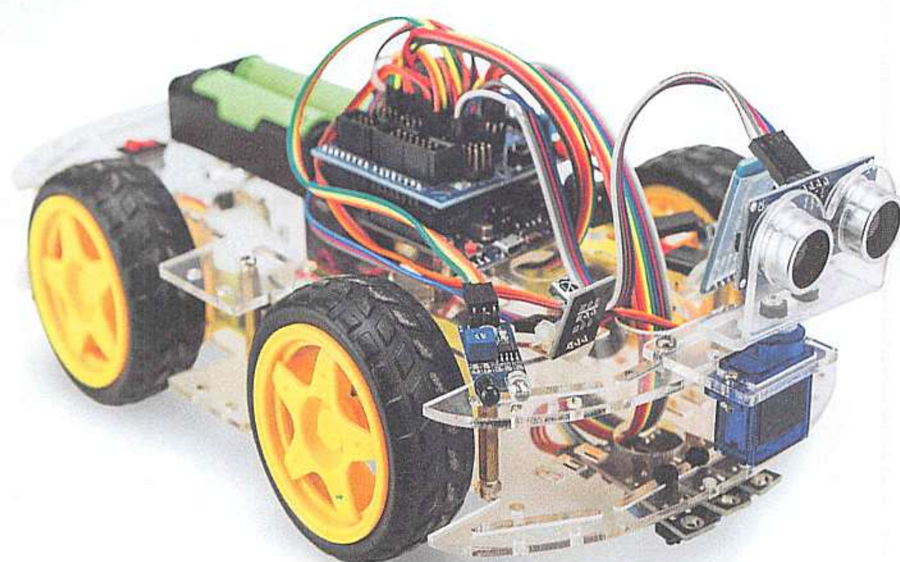
Proteus (Circuit Simulator for Professional) ต่อ



▲ ตัวอย่าง Library ของ Arduino, Ultrasonic, LCD, SIM900D, Gas Sensor, RTC ของ Proteus

บทสรุปท้ายบท

ซอฟต์แวร์ที่แนะนำในบทที่ 4 เป็นเพียงการสาธิตตัวอย่างคร่าวๆ ให้พอเห็นภาพเท่านั้น ไม่ได้ถ่ายทอดในแบบคู่มือสอนการใช้งานอย่างละเอียด ผู้อ่านจะต้องหาคู่มือสอนการใช้งานมาศึกษาด้วยตนเอง หรือศึกษาจาก Tutorial ของเว็บไซต์ผู้ผลิตซอฟต์แวร์โดยตรง การนำเสนอในบทนี้เป็นเพียง Guideline เพื่อแนะนำให้รู้จักและทดลองใช้งานซอฟต์แวร์ หรือแพลตฟอร์มออนไลน์โดยไม่จำเป็นต้องเสียค่าใช้จ่าย เพื่อให้มีโอกาส Simulation ก่อนไปซื้ออุปกรณ์จริงมาต่อวงจรสร้างโครงงานหรือพัฒนาสิ่งประดิษฐ์ขึ้นใช้เองที่บ้านหรือที่ทำงาน อย่างไรก็ตาม ยังมีซอฟต์แวร์อีกหลายตัวที่น่าสนใจ ลองค้นหาและทดลองเล่นดูครับ...



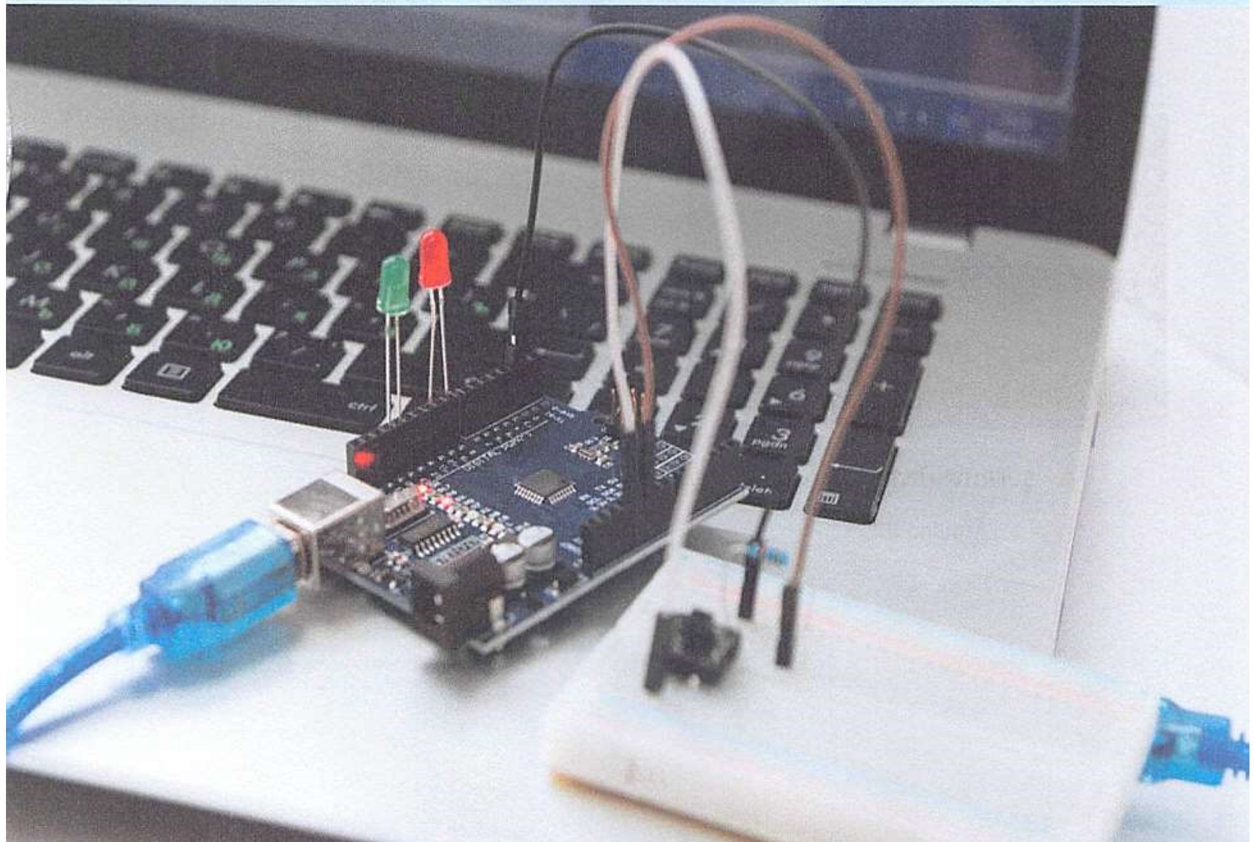
CHAPTER

05

เริ่มใช้งานบอร์ด Arduino ครั้งแรก

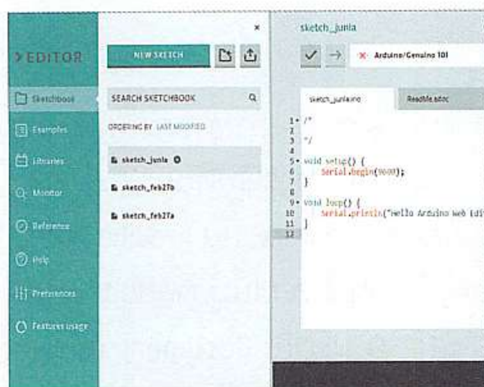
How to get started with Arduino

เมื่อเราตัดสินใจซื้อบอร์ด Arduino และได้มันมาไว้ในมือแล้ว แต่หลายคนคงติดปัญหาเดียวกัน คือ ไม่รู้จะเริ่มต้นอย่างไรกับบอร์ด? บทนี้คือวิธีการเริ่มต้นใช้งานบอร์ด Arduino ครอบคลุมการติดตั้งซอฟต์แวร์ การเชื่อมต่อ Arduino IDE กับ Arduino Board ต่อด้วยการเรียนรู้ก้าวแรกผ่าน Built-in Examples โดยในขั้นตอนนี้ผู้อ่านยังไม่จำเป็นต้องมีทักษะอะไรมากนัก แค่เปิดดูตัวอย่างและอ่านทำความเข้าใจ Document Project ต่อวงจรตามแบบ โหลดโค้ดลงบอร์ด สังเกตผลลัพธ์ และทำซ้ำกับตัวอย่างอื่นๆ



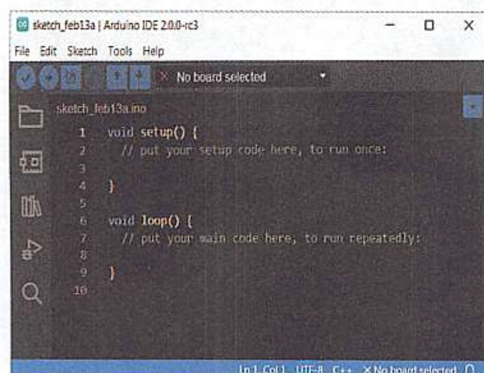
ทางเลือกในการเขียนโปรแกรม

บอร์ด Arduino นั้นจะทำอะไรไม่ได้เลยหากไม่มีชุดคำสั่งควบคุมการทำงาน โดยชุดคำสั่งนี้เราจะต้องเขียนขึ้นมาจากคอมพิวเตอร์แล้วโหลดเข้าไปที่บอร์ด เพื่อให้บอร์ดทำงานตามที่เราต้องการ ไม่ต่างจากคอมพิวเตอร์ซึ่งเป็นเพียงฮาร์ดแวร์ หากปราศจากซอฟต์แวร์แล้วมันก็หมดประโยชน์ ในการเขียนชุดคำสั่งหรือโค้ดดิง (Coding) เราเลือกได้ 2 ทาง ดังนี้



▲ รูปแสดงเว็บเพจ Arduino Web Editor

1. **Arduino Web Editor (Coding Online)** คือ การเขียนโปรแกรมผ่านทางออนไลน์ หรือ Arduino Cloud ในเว็บไซต์ arduino.cc กรณีนี้ต้องเชื่อมต่อกับอินเทอร์เน็ตในการเข้าใช้งานโปรแกรมแบบ Online IDE ไม่ต้องติดตั้งโปรแกรมลงเครื่อง เราสามารถเขียนโปรแกรมผ่านทาง Arduino Web Editor ได้โดยคลิกที่ Code Online แล้ว Login เพื่อเข้าใช้งาน จะปรากฏหน้าจอดังรูป



▲ รูปแสดงหน้าต่างโปรแกรม Arduino IDE

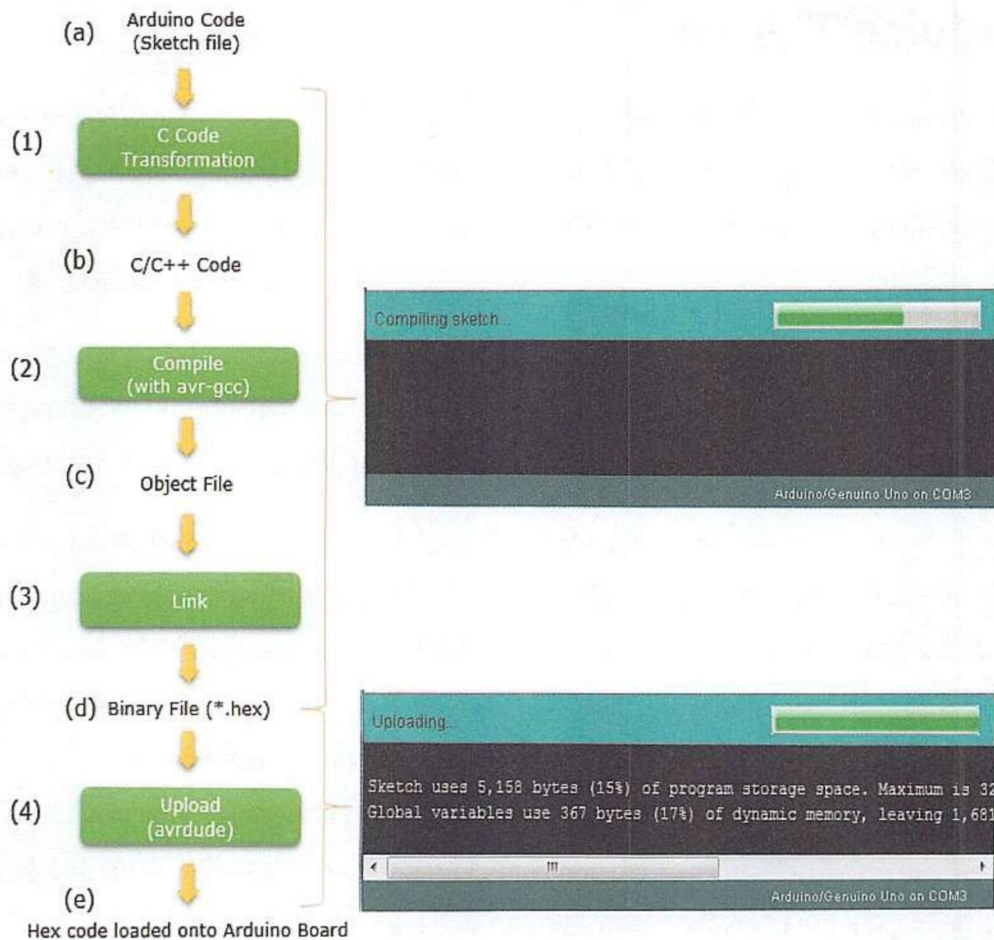
2. **Arduino IDE (Coding Offline)** เป็นการเขียนโปรแกรมบนคอมพิวเตอร์ของเราโดยไม่ต้องต่ออินเทอร์เน็ต กรณีนี้ต้องดาวน์โหลดและติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์ก่อน โดยเลือกรุ่นของ Arduino IDE ตามระบบปฏิบัติการของเครื่องคอมพิวเตอร์ที่เราใช้ จากนั้นติดตั้งโปรแกรมตามขั้นตอน และเปิดโปรแกรมจะปรากฏหน้าจอดังรูป

Arduino Sketch

Arduino Sketch ถ้าจะให้คำจำกัดความง่ายๆ ก็คือ ตัวโปรเจ็กต์ซึ่งเขียนอยู่ในโปรแกรม Arduino IDE หรือ Arduino Web Editor ที่มีไว้สำหรับเขียนโปรแกรมหรือโค้ดที่มนุษย์อ่านได้ ซึ่งในที่นี้ก็คือ ภาษา C/C++ (Human-Readable Code) แล้วอัพโหลดไปยังบอร์ด Arduino เพื่อสั่งให้ทำงานตามต้องการ ตัวโปรแกรมหรือโค้ดที่เขียนขึ้นในโปรเจ็กต์นี้เราจะเรียกว่า **"สเกตซ์ (Sketch)"** ซึ่งจะถูกเก็บอยู่ใน Sketchbook

ภาษาโปรแกรมบน Arduino จะมีลักษณะคล้ายกับการเขียนโปรแกรมด้วยภาษา C/C++ เนื่องจากพัฒนาด้วย Framework แบบ Simplified Version ของภาษา C/C++ จึงทำให้คุ้นเคยและใช้งานง่าย เราสามารถเรียนรู้การเขียนโปรแกรมแบบเข้มข้นได้ใน Chapter 6 พื้นฐานภาษา C และการ Coding

แต่ก่อนที่จะอัพโหลดตัว Sketch ไปยัง Arduino Board ไม่ว่าจะใช้ Arduino IDE หรือ Arduino Web Editor ก็ตาม ยังต้องมีหลายขั้นตอนเพื่อสร้างแอปพลิเคชัน โดยขั้นตอนแรกของการสร้าง Sketch คือการดำเนินการแปลงโค้ดล่วงหน้า (Preprocessing หรือ C Code Transformation) เพื่อทำให้โค้ด Arduino หรือไฟล์ Sketch (.ino) กลายเป็นโปรแกรมภาษา C/C++ และถูกส่งผ่านไปยัง Compiler ทำหน้าที่ Compiling เปลี่ยนรหัสภาษา C/C++ ซึ่งเป็นภาษาที่มนุษย์อ่านได้ไปเป็นภาษาที่คอมพิวเตอร์อ่านได้ในรูปแบบ Object Files (.a) โดย Object Files เหล่านี้จะ Link กับไลบรารีของ Arduino (Arduino Library) ที่รวบรวมคำสั่งพื้นฐานต่างๆ สำหรับใช้งาน Arduino ไว้ ผลลัพธ์ของการ Link นี้จะได้ Binary File ที่เป็นไฟล์เลขฐานสิบหก (.hex) ที่สามารถอัพโหลดตัว Sketch ไปยัง Arduino Board เพื่อสั่งให้ทำงานได้



▲ กระบวนการสร้างและอัพโหลดไฟล์ มีขั้นตอนที่เกิดขึ้นเมื่อคลิกปุ่ม Run บน Arduino IDE

Credit : http://www.sharetechnote.com/html/Arduino_Programming.html

ต่อไปจะเป็นการเรียนรู้ 2 เครื่องมือที่ช่วยให้เราสามารถเขียนโปรแกรมและสร้าง Sketch สำหรับสั่งงาน Arduino กัน โดยมีสิ่งที่ต้องเตรียมดังต่อไปนี้

1. บอร์ด Arduino Uno
2. สาย USB แบบ B Connection
3. คอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการ ได้แก่ Windows 10/8/7, Mac หรือ Linux OS
4. ซอฟต์แวร์ Arduino IDE

การเขียนโปรแกรมบน Arduino IDE ครั้งแรก

Step 1 ดาวน์โหลดและติดตั้ง Arduino IDE

Arduino IDE เป็นโปรแกรมใช้เขียนโปรแกรม หรือชุดคำสั่งที่สามารถแปลงภาษาที่มนุษย์เข้าใจ (Human-Readable) ซึ่งในที่นี้คือ ภาษา C/C++ เป็นภาษาที่คอมพิวเตอร์เข้าใจ (Machine-Readable) ผ่านตัวคอมไพเลอร์ (Compiler) แล้วอัพโหลดไฟล์โปรแกรมที่เขียนลงบนบอร์ด Arduino เพื่อให้มันทำงานตามคำสั่งที่เราได้เขียนโปรแกรมไว้



▲ เว็บเพจดาวน์โหลดโปรแกรม Arduino IDE

https://www.arduino.cc/en/software

เมื่อดาวน์โหลดและติดตั้งโปรแกรม Arduino IDE เสร็จแล้ว เมื่อเปิดใช้งานครั้งแรกจะปรากฏหน้าต่างโปรแกรมเหมือนกันแม้จะรันบนระบบปฏิบัติการที่แตกต่างกัน ดังรูป

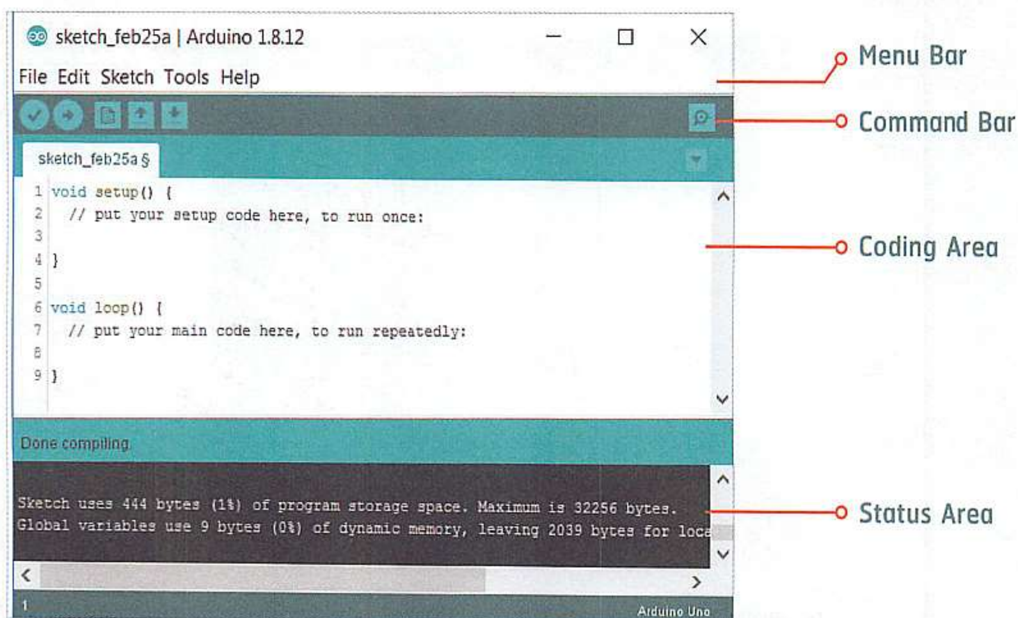


▲ หน้าตาโปรแกรม Arduino IDE

Step 2 รู้จักส่วนประกอบของโปรแกรม Arduino IDE

ก่อนจะเริ่มเรียนรู้การใช้งานโปรแกรม Arduino IDE กันในช่วงท้ายบทเพื่อสร้าง Sketch แรกของเรา และจะสอนการเขียนโปรแกรมบน Arduino Web Editor ควบคู่กันไป ในตอนนี้จะขอเริ่มด้วยการพาผู้อ่านทุกท่านสำรวจหน้าต่างโปรแกรมกันเสียก่อนว่า ส่วนไหนเรียกว่าอะไร เมนูคำสั่งมีอะไรบ้าง และสัญลักษณ์ต่างๆ ใช้แทนเครื่องมืออะไร

โปรแกรม Arduino IDE ประกอบด้วย 4 ส่วนหลัก ดังนี้



▲ ส่วนประกอบหน้าต่างโปรแกรม Arduino IDE

1. **Menu Bar** คือ เมนูสำหรับทำหน้าที่บริหารจัดการข้อมูลของไฟล์ Sketch หรือไฟล์โค้ดของโปรแกรม

1.1 แถบเมนูคำสั่ง File : ใช้สำหรับจัดการไฟล์ Sketch

File Edit Sketch Tools Help

a.	New	Ctrl+N
b.	Open...	Ctrl+O
c.	Open Recent	>
d.	Sketchbook	>
e.	Examples	>
f.	Close	Ctrl+W
g.	Save	Ctrl+S
h.	Save As...	Ctrl+Shift+S
i.	Page Setup	Ctrl+Shift+P
j.	Print	Ctrl+P
k.	Preferences	Ctrl+Comma
l.	Quit	Ctrl+Q

a. สร้างไฟล์ Sketch ใหม่
b. เปิดไฟล์ Sketch ที่บันทึกไว้ในโฟลเดอร์
c. เปิดไฟล์ Sketch ที่เคยเปิดเมื่อไม่นานมานี้
d. เปิดไฟล์ Sketch ล่าสุดที่อยู่ในโฟลเดอร์ Documents\Arduino
e. ตัวอย่างโค้ดสำหรับทดลองอัพโหลดลงบอร์ด
f. ปิดไฟล์ Sketch ที่เปิดอยู่
g. บันทึกไฟล์ Sketch ปัจจุบัน
h. บันทึกไฟล์ Sketch โดยเปลี่ยนชื่อไฟล์
i. ตั้งค่าหน้ากระดาษของไฟล์งานปัจจุบัน
j. สั่งพิมพ์งานออกทางเครื่องพิมพ์
k. ตั้งค่าการทำงานต่างๆ ของโปรแกรม
l. ปิดและออกจากโปรแกรม

1.2 แถบเมนูคำสั่ง Edit : ใช้สำหรับการแก้ไขโปรแกรม Sketch ที่เขียนขึ้น

Edit Sketch Tools Help

a.	Undo	Ctrl+Z
b.	Redo	Ctrl+Y
c.	Cut	Ctrl+X
d.	Copy	Ctrl+C
e.	Copy for Forum	Ctrl+Shift+C
f.	Copy as HTML	Ctrl+Alt+C
g.	Paste	Ctrl+V
h.	Select All	Ctrl+A
i.	Go to line...	Ctrl+L
j.	Comment/Uncomment	Ctrl+Slash
k.	Increase Indent	Tab
l.	Decrease Indent	Shift+Tab
m.	Increase Font Size	Ctrl+Plus
n.	Decrease Font Size	Ctrl+Minus
o.	Find...	Ctrl+F
p.	Find Next	Ctrl+G
q.	Find Previous	Ctrl+Shift+G

a. ย้อนกลับไปคำสั่งหรือการพิมพ์ครั้งที่แล้ว
b. ย้อนกลับคืนเมื่อคลิกเลือก Undo หรือต้องการกลับสู่คำสั่งล่าสุด
c. ตัดข้อความที่ต้องการคัดลอกไว้ไปเก็บใน Clipboard
d. คัดลอกข้อความที่เลือกไว้มาเก็บใน Clipboard
e. คัดลอก Code ทั้งหมดเก็บไว้ใน Clipboard
f. คัดลอก Code ทั้งหมดแบบ HTML เก็บไว้ใน Clipboard
g. วางข้อความที่ตัดหรือคัดลอกมา
h. เลือกข้อความทั้งหมด
i. กระโดดเคอร์เซอร์ไปยังบรรทัดที่ต้องการ
j. ใส่เครื่องหมาย // เพื่อทำการ Comment/UnComment
k. เลื่อนข้อความทั้งบรรทัดไปข้างหน้า
l. เลื่อนข้อความทั้งบรรทัดไปข้างหลัง
m. ขยายขนาดตัวอักษรทั้งโปรแกรม
n. ลดขนาดตัวอักษรทั้งโปรแกรม
o. ค้นหาคำในโปรแกรม
p. ค้นหาคำถัดไปในโปรแกรม โดยต้องใช้คำสั่ง Find ค้นหาคำก่อน
q. ค้นหาคำย้อนกลับไปในโปรแกรม โดยต้องใช้คำสั่ง Find ค้นหาคำก่อน

1.3 แถบเมนูคำสั่ง Sketch : ใช้สำหรับตรวจสอบ คอมไพล์ อัพโหลด และเพิ่ม Library ให้กับการเขียนโปรแกรม

Sketch Tools Help

a.	Verify/Compile	Ctrl+R
b.	Upload	Ctrl+U
c.	Upload Using Programmer	Ctrl+Shift+U
d.	Export compiled Binary	Ctrl+Alt+S
e.	Show Sketch Folder	Ctrl+K
f.	Include Library	›
g.	Add File...	

a. ตรวจสอบและคอมไพล์
b. อัพโหลดไฟล์ Sketch ที่เขียนขึ้นลงบอร์ด Arduino
c. อัพโหลดไฟล์ Sketch ผ่านเครื่องโปรแกรม เช่น Arduino as ISP หรือ USBASP เป็นต้น
d. Export ไฟล์ Binary (.hex) ที่คอมไพล์แล้วเก็บใน Sketch Folder
e. เปิด Folder ที่เก็บไฟล์ Sketch หรือไฟล์โปรแกรม
f. เรียกใช้ Library (Header File: .h) สำหรับใช้คำสั่งและเชื่อมต่อกับอุปกรณ์ต่างๆ ของ Arduino เพิ่มเติม
g. เพิ่ม File ให้กับ Sketchbook ปัจจุบัน

1.4 แถบเมนูคำสั่ง Tools : ใช้สำหรับเป็นเครื่องมือช่วยในการใช้คำสั่งต่างๆ ของ Arduino

Tools Help

a.	Auto Format	Ctrl+T
b.	Archive Sketch	
c.	Fix Encoding & Reload	
d.	Manage Libraries...	Ctrl+Shift+I
e.	Serial Monitor	Ctrl+Shift+M
f.	Serial Plotter	Ctrl+Shift+L
g.	WiFi101 / WiFiNINA Firmware Updater	
h.	ArduBlock	
i.	Board: "Arduino/Genuino Uno"	›
j.	Port	›
k.	Get Board Info	
l.	Programmer: "AVRISP mkII"	›
m.	Burn Bootloader	

a. จัดรูปแบบการเขียนโปรแกรมให้ดูเหมาะสมแบบอัตโนมัติ
b. บีบอัดไฟล์ (.Zip format) ทั้ง Folder ของไฟล์ Sketch
c. การ Fix การเข้ารหัส และโหลดซ้ำใหม่
d. การจัดการ Library ต่างๆ ของ Arduino เช่น การ Install Library
e. เปิดหน้าต่าง Serial Monitor เพื่อตรวจสอบ รับส่งข้อมูล และดูการสื่อสารผ่าน Serial Port (สามารถคลิกที่สัญลักษณ์แว่นขยายที่มีเครื่องหมายจุดในหน้าต่างโปรแกรมได้เช่นกัน)
f. รับค่าจาก Serial Port มา Plot กราฟ
g. การอัพเดตเฟิร์มแวร์ของ WiFi Module
h. เพิ่มเติมเครื่องมือ ArduBlock สำหรับช่วยในการเขียนโปรแกรมแบบ Block-based Programming (เพิ่มเติมได้เองภายหลังการติดตั้งโปรแกรม Arduino IDE)
i. เลือกรุ่นบอร์ด Arduino ให้ตรงกับที่ใช้ในการ Upload โปรแกรม
j. เลือกหมายเลข Port ที่ใช้เชื่อมต่อเครื่องคอมพิวเตอร์กับบอร์ด Arduino ให้ตรงกัน เช่น COM1
k. แสดงข้อมูลของบอร์ด
l. เลือกตัวโปรแกรมเมอร์ เช่น AVRISP mkII, Arduino as ISP เป็นต้น
m. เบิร์นตัว Bootloader

1.5 **แถบเมนูคำสั่ง Help :** ใช้เมื่อต้องการความช่วยเหลือ หรืออยากได้ข้อมูลที่เกี่ยวข้องกับโปรแกรม

Help
- a. Getting Started
- b. Environment
- c. Troubleshooting
- d. Reference
- e. Galileo Help
  - Getting Started
  - Troubleshooting
- f. Edison Help
  - Getting Started
  - Troubleshooting
- g. Find in Reference Ctrl+Shift+F
- h. Frequently Asked Questions
- i. Visit Arduino.cc
- j. About Arduino

a. เอกสารสอนการเริ่มต้นใช้งานโปรแกรม Arduino IDE และ Arduino Board
b. รายละเอียดของเมนูต่างๆ ในโปรแกรม Arduino Software (IDE)
c. รวมคำถามและวิธีการแก้ไขปัญหาต่างๆ ของการใช้งาน Arduino
d. แสดง Language Reference หรือคำสั่งต่างๆ สำหรับการเขียนภาษาโปรแกรม
e. รายละเอียดการใช้งานบอร์ด Intel Galileo และ Arduino IDE ประกอบด้วย Getting Started และ Troubleshooting
f. รายละเอียดการใช้งานบอร์ด Intel Edison และ Arduino IDE ประกอบด้วย Getting Started และ Troubleshooting
g. ค้นหาข้อมูลในเอกสาร Reference
h. คำถามที่ถามบ่อยที่เกี่ยวกับการใช้งาน Arduino
i. ไปที่เว็บไซต์ของ Arduino.cc
j. ข้อมูลเกี่ยวกับโปรแกรม Arduino IDE เช่น การแสดง Version

2. **Command Bar** คือ คำสั่งต่างๆ ที่สำหรับใช้ตรวจสอบ คอมไพล์ อัพโหลด สร้างไฟล์ใหม่ เปิดไฟล์ และบันทึกไฟล์ Sketch ในการเขียนโปรแกรม



a. คำสั่ง **Verify** ใช้ตรวจสอบความถูกต้องของโปรแกรมที่เขียนขึ้น
b. คำสั่ง **Upload** ใช้ตรวจสอบความถูกต้องของโปรแกรม (Verify) และอัพโหลดลงบอร์ด Arduino
c. คำสั่ง **New** ใช้สร้างไฟล์โปรแกรมใหม่
d. คำสั่ง **Open** ใช้เปิดไฟล์โปรแกรม
e. คำสั่ง **Save** ใช้บันทึกไฟล์โปรแกรม
f. คำสั่ง **Serial Monitor** เปิดหน้าต่างสำหรับตรวจสอบ รับส่งข้อมูล และแสดงผลข้อมูลผ่าน Serial Port

3. **Coding Area** คือ พื้นที่ใช้เขียนโปรแกรมสำหรับ Arduino โดยใน Arduino IDE จะมีฟังก์ชัน setup() สำหรับเขียนโปรแกรมตั้งค่าอุปกรณ์ และ loop() สำหรับเขียนโปรแกรมควบคุมและสั่งการให้บอร์ด Arduino ทำงานตามต้องการ ซึ่งทั้ง 2 ฟังก์ชันจะถูกสร้างขึ้นโดยอัตโนมัติ

```
sketch_feb25a

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

▲ ภาพแสดงพื้นที่สำหรับเขียนโค้ด

4. **Status Area** คือ พื้นที่แสดงสถานะเพื่อแจ้งให้ทราบว่าเกิดอะไรขึ้นเมื่อ Arduino IDE ทำอะไรบางอย่าง เช่น การคอมไพล์ การอัพโหลด หรือการตรวจสอบโค้ดที่เขียนขึ้น

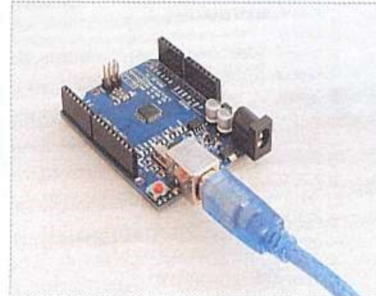
```
Done compiling.

Sketch uses 444 bytes (1%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
1                                                                                Arduino Uno
```

▲ ภาพแสดงพื้นที่แจ้งสถานะการทำงาน ณ ขณะนั้น

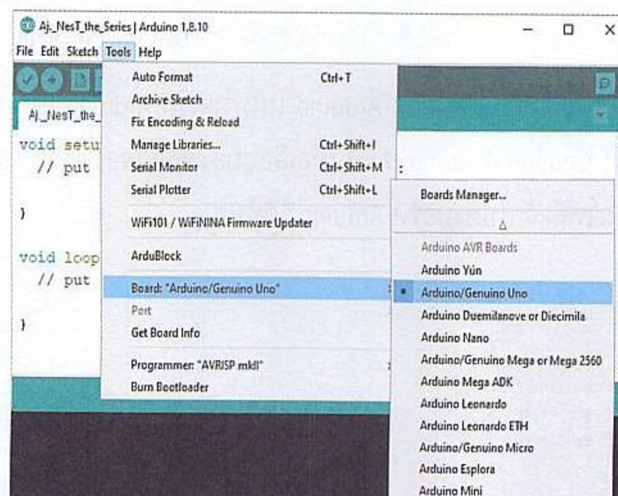
Step 3 ตั้งค่าการเชื่อมต่อ Arduino IDE กับบอร์ด

การอัพโหลดโค้ด หรือ Sketch ไปยังบอร์ด Arduino จำเป็นต้องต่อสาย USB เชื่อมระหว่างบอร์ด Arduino กับเครื่องคอมพิวเตอร์ที่ติดตั้ง Arduino IDE เรียบร้อยแล้ว หลังจากนั้นให้ตั้งค่าโดยกำหนดรุ่นบอร์ด Arduino ที่ใช้งาน และเลือก Port ที่ใช้เชื่อมต่อกับบอร์ด Arduino ให้ถูกต้อง



Step 3.1 กำหนดรุ่นของบอร์ดที่ใช้ให้กับ IDE

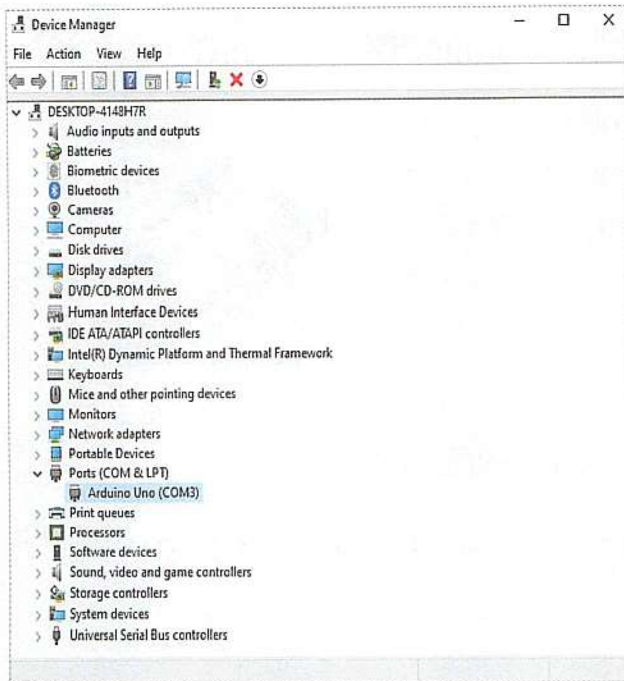
วิธีการตั้งค่า ให้คลิกเมนู **Tools > Board > เลือกรุ่นของบอร์ด Arduino** ที่ต้องการใช้งานให้ถูกต้อง ในที่นี้ใช้รุ่น **Arduino/Genuino Uno**



▲ การตั้งค่ารุ่นของบอร์ด Arduino เพื่อบอกกับ Arduino IDE ว่าเราใช้บอร์ดรุ่นไหน

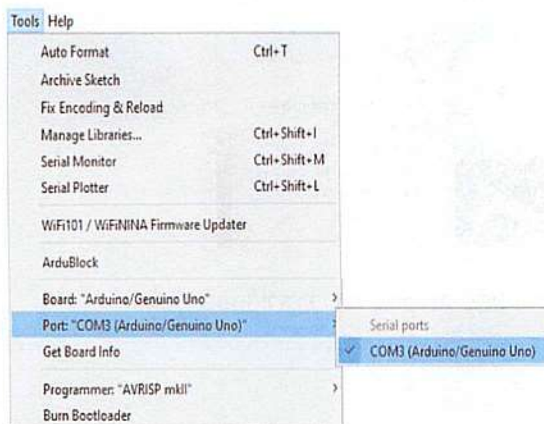
Step 3.2 ตั้งค่า Port ที่ใช้ติดต่อกับบอร์ด

จากนั้นตั้งค่า Port โดยต้องรู้ว่า Arduino IDE เชื่อมต่อกับบอร์ด Arduino ที่ Port หมายเลขใด วิธีการดูว่าใช้ Port อะไร ถ้าใช้ระบบปฏิบัติการ Windows ให้เข้าไปที่ **Device Manager > Ports (COM & LPT)** จะปรากฏข้อความว่า **"Arduino Uno (COM3)"** แสดงว่าใช้ Port: COM3 ในการเชื่อมต่อกับบอร์ด Arduino



◀ วิธีการดูว่า Port อะไรในการเชื่อมต่อกับบอร์ด Arduino

ตั้งค่าการใช้งาน Port ที่เชื่อมต่อบอร์ด Arduino ในโปรแกรม Arduino IDE ให้คลิกเมนู Tools > Port ในที่นี้คือ COM3 (Arduino/Genuino Uno) เมื่อเลือกบอร์ดและพอร์ตแล้ว Arduino IDE ก็พร้อมที่จะอัพโหลดโค้ดที่คอมไพล์แล้วไปยังบอร์ด Arduino ได้ทันที

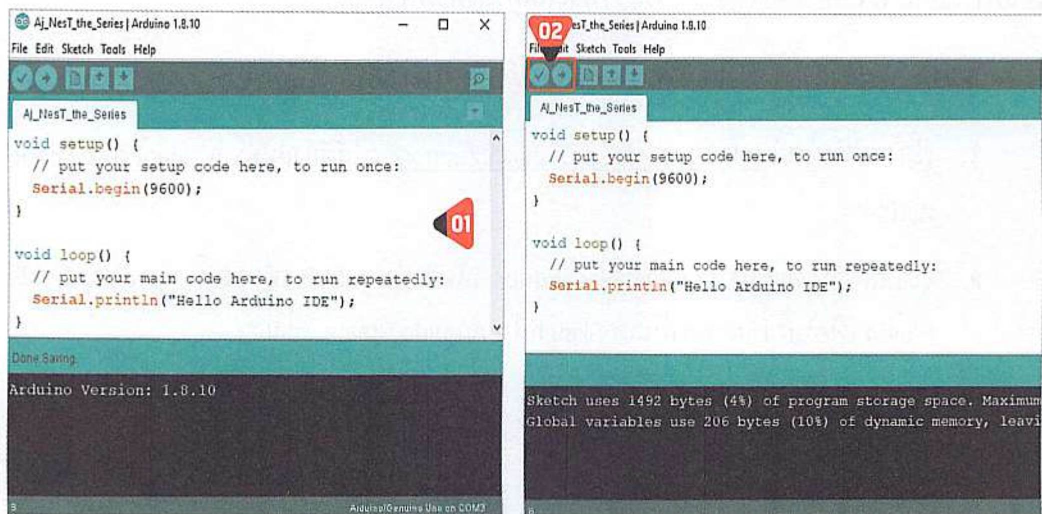


◀ วิธีการตั้งค่า Port เชื่อมต่อบอร์ด Arduino

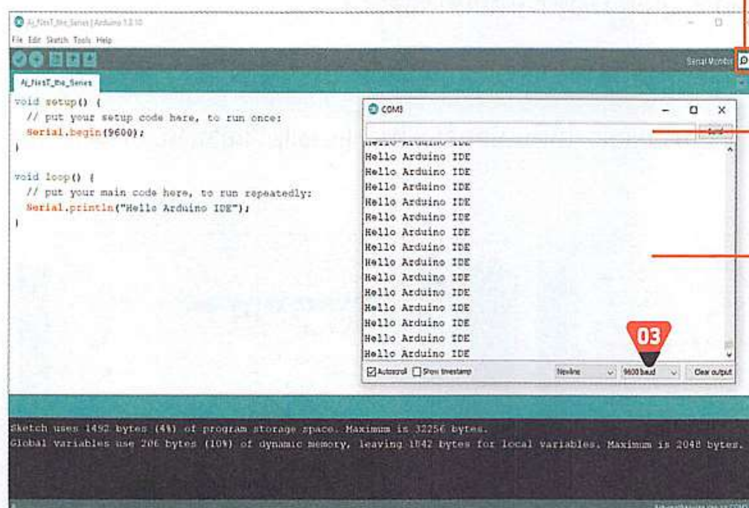
ถ้ายังไม่มีการต่อสาย USB เชื่อมระหว่างบอร์ด Arduino กับคอมพิวเตอร์ จะคลิกเมนู Tools > Port ไม่ได้ เพราะสถานะจะยังไม่ Active นั่นเอง

Step 3.3 ทดลองเขียนโปรแกรมบน Arduino IDE ครั้งแรก

1. เขียนโปรแกรมแสดงผลออกทางหน้าจอ Serial Monitor
2. คลิกปุ่ม Verify และ Upload



3. เปิด Serial Monitor กำหนดค่า Baud Rate เป็น 9600 baud ผลลัพธ์หน้าจอจะแสดงคำว่า Hello Arduino IDE ขึ้นบรรทัดใหม่ และพิมพ์ซ้ำไปเรื่อยๆ

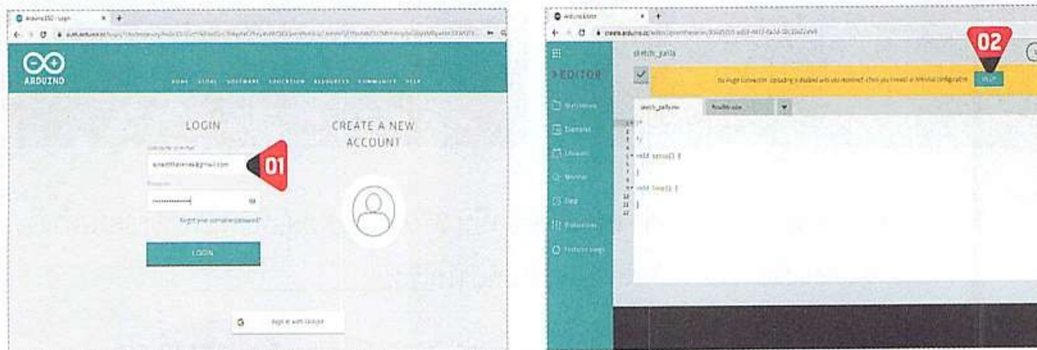


การเขียนโปรแกรมบน Arduino Web Editor ครั้งแรก

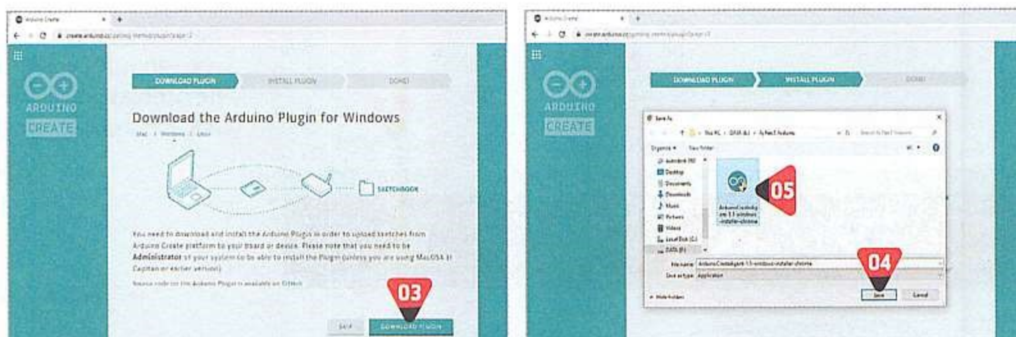
Arduino Web Editor เป็นพื้นที่สำหรับเขียนโค้ดและอัพโหลด Sketche เข้าไปที่บอร์ด Arduino ได้จากเว็บเบราว์เซอร์ เช่น Chrome, Firefox, Safari และ Edge โดยทางเว็บแนะนำให้ใช้ Google Chrome เป็นการเขียนโค้ดแบบไม่ต้องติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์

Step 1 การใช้งาน Arduino Web Editor

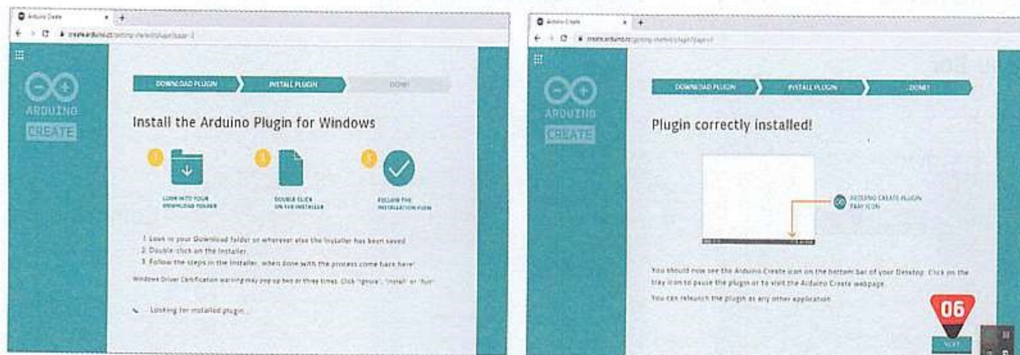
1. เข้าไปที่เว็บไซต์ https://create.arduino.cc/editor/ ผู้ใช้ต้องสมัคร Account และ Log in เข้าใช้งาน
2. จะแสดงเว็บเพจของ Arduino Web Editor ให้คลิกปุ่ม **HELP** เพื่อติดตั้ง Arduino Create Plugin เพื่อรองรับการอัพโหลด Sketch ใน Arduino Create Editor



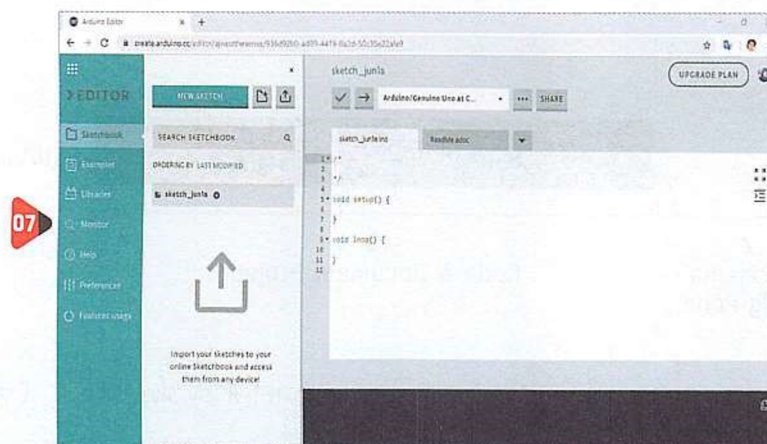
3. คลิกปุ่ม **DOWNLOAD PLUGIN** เพื่อดาวน์โหลดไฟล์ติดตั้ง
4. เลือกตำแหน่งเก็บไฟล์ติดตั้งลงเครื่อง แล้วคลิกปุ่ม **Save**
5. เมื่อดาวน์โหลดเสร็จเรียบร้อยแล้ว ให้ดับเบิลคลิกที่ไฟล์ **Installer** แล้วทำตามขั้นตอนของ **Installer**



6. เมื่อติดตั้งเสร็จจะพบไอคอน **Arduino Create** ปรากฏที่ System Tray ตรงมุมล่างด้านขวาของจอภาพ คลิกปุ่ม **NEXT** เป็นอันเสร็จสมบูรณ์

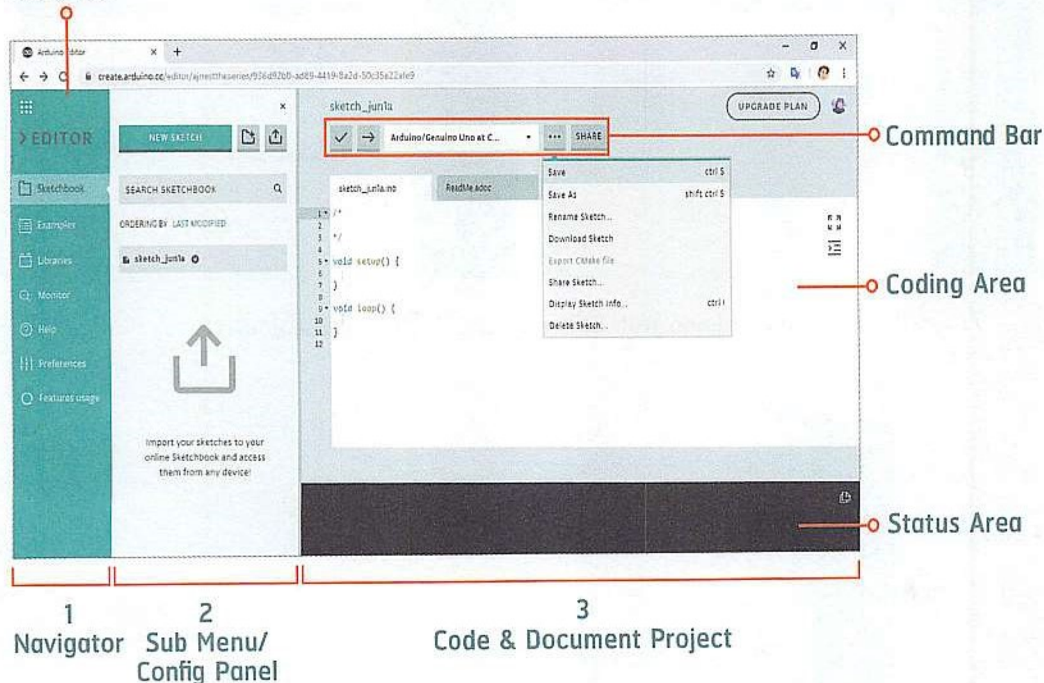


7. เข้าสู่หน้าเว็บเพจ **Arduino Web Editor** พร้อมสำหรับการเขียนโปรแกรม



Step 2 รู้จักส่วนประกอบของ Arduino Web Editor

โปรแกรม Arduino Web Editor จะประกอบด้วย 3 ส่วน ดังนี้

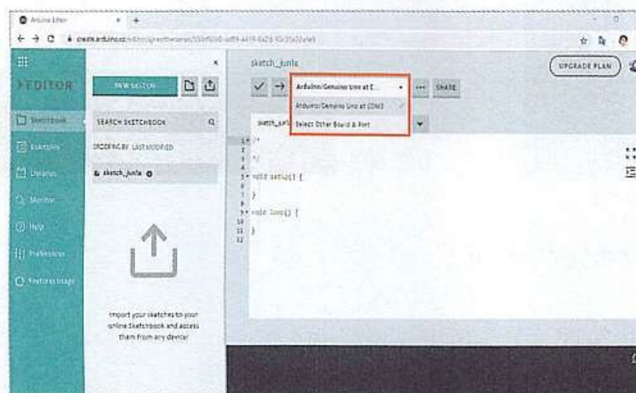


Menu Bar	แถบเมนูหลักที่เป็นเหมือน Navigator ประกอบด้วย Sketchbook, Examples, Libraries, Serial Monitor, Help, Preferences และ Features Usage
Command Bar	แถบคำสั่งของ Web Editor ประกอบด้วย Verify, Upload, Select Board or Port, Save, Save As, Rename Sketch..., Download Sketch, Export CMake File, Share Sketch..., Display Sketch Info..., Delete Sketch... และการแชร์ (Share) โค้ดแบบออนไลน์ได้
Coding Area	พื้นที่เขียนโค้ดของ Web Editor จะเหมือนกับ Arduino IDE ที่ประกอบด้วยฟังก์ชัน setup() และฟังก์ชัน loop() ที่ถูกสร้างขึ้นอัตโนมัติ
Status Area	พื้นที่สถานะของ Web Editor จะเหมือนกับ Arduino IDE ที่ทำให้เราทราบสถานะว่ากำลังเกิดอะไรขึ้น ณ ขณะนี้ เช่น Compiling, Uploading หรือ Verifying โค้ด

Step 3 ตั้งค่าเชื่อมต่อ Arduino Web Editor กับบอร์ด

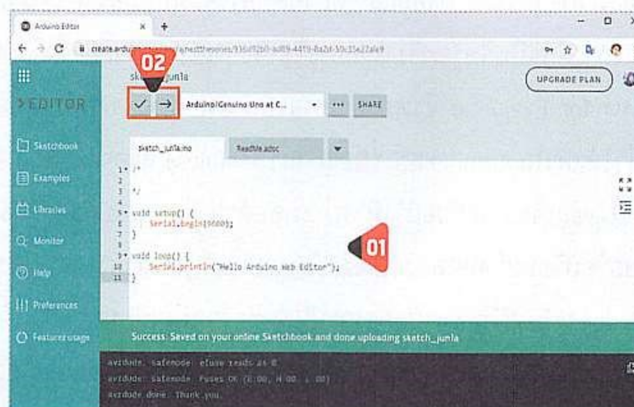
Step 3.1 กำหนดรุ่นของบอร์ดและการใช้งาน Port

ให้ไปที่หน้าเว็บเพจ Arduino Web Editor แล้วต่อสาย USB เชื่อมบอร์ด Arduino เข้ากับคอมพิวเตอร์ บอร์ดจะได้รับไฟเลี้ยงและพร้อมทำงานทันที ปกติเราจะต้องตั้งค่าให้บอร์ดกับคอมพิวเตอร์รู้จักกัน และกำหนด Port ที่จะใช้สื่อสาร แต่ถ้าใช้บอร์ดรุ่นที่โปรแกรมซัพพอร์ตอยู่แล้ว มันก็จะตั้งค่าให้อัตโนมัติ ซึ่งผู้ใช้สามารถเลือกเพิ่มเติมได้เองเช่นกัน ในตัวอย่างนี้จะเห็นว่ามีเครื่องหมายถูกปรากฏที่ Arduino/Genuino Uno at COM3 แสดงถึงรุ่นและพอร์ตที่มันค้นเจอ ถึงตอนนี้เราก็สามารถติดต่อและอัพโหลดโค้ดได้แล้ว

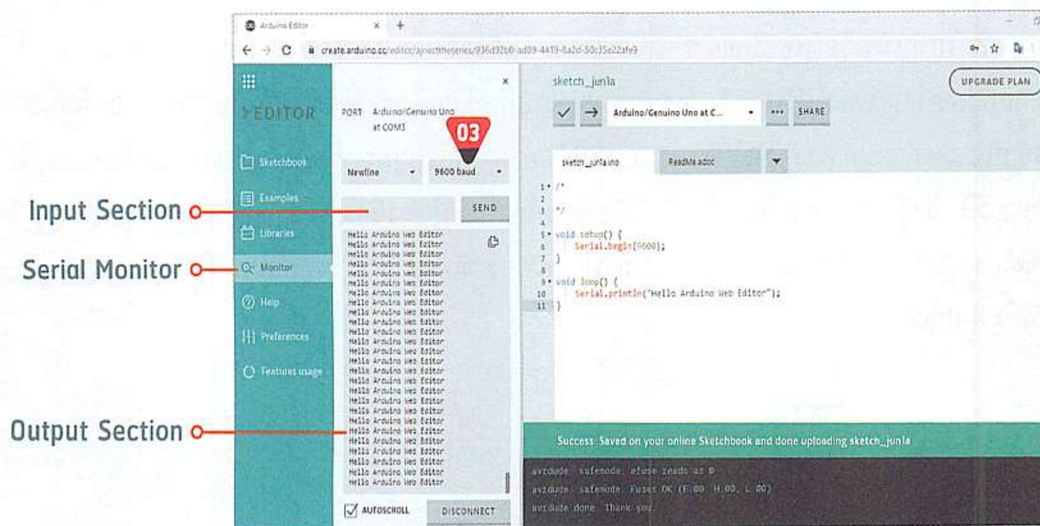


Step 3.2 ทดลองเขียนโปรแกรมบน Arduino Web Editor ครั้งแรก

1. เขียนโปรแกรมแสดงผลออกทางหน้าจอ Serial Monitor (พิมพ์โค้ดบรรทัดที่ 6, 10 ตามรูป)
2. จากนั้นคลิกปุ่ม Verify และ Upload (ถ้าไม่ต่อบอร์ดมันจะแสดงข้อความว่า Error uploading)



3. คลิกเมนู **Monitor** ใส่ค่า **Baud Rate** เป็น **9600 baud** หน้าจอจะแสดงคำว่า Hello Arduino Web Editor ขึ้นบรรทัดใหม่ และพิมพ์ซ้ำไปเรื่อยๆ



Input Section	ส่วนรับค่า Input
Serial Monitor	เปิดใช้สำหรับรับค่าและแสดงผล
Output Section	ส่วนแสดงผล Output ในการทดลองนี้ใช้เฉพาะส่วนแสดงผล

การศึกษา Arduino ในภาคปฏิบัติ

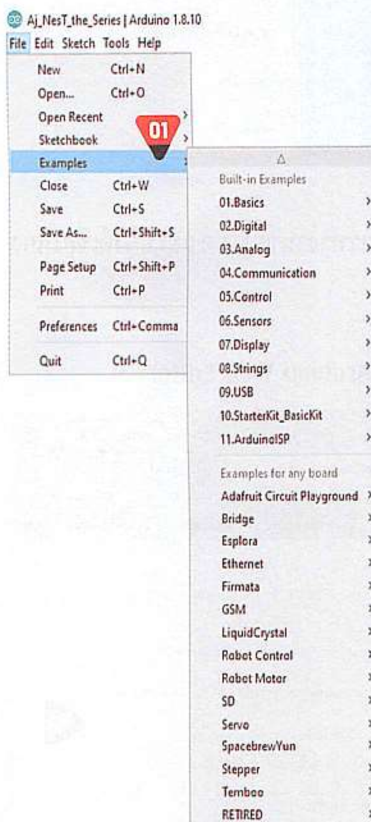
เมื่อติดตั้งโปรแกรม IDE เสร็จเรียบร้อยแล้ว หลายคนอาจจะพบอุปสรรค เช่น ยังขาดประสบการณ์การต่อวงจร ยังเขียนภาษา C ไม่เป็น ไม่รู้จะเริ่มต้นอย่างไร จริงๆ แล้วทาง Arduino มีเจตนารมณ์ที่ชัดเจนว่า Arduino ถูกสร้างขึ้นมา for Everyone จึงพยายามทลายกำแพงทุกอย่างที่เป็นอุปสรรคต่อการเรียนรู้ โดยใส่ตัวอย่างง่ายๆ ไว้ให้แล้วใน Arduino IDE (Built-in Examples) หรือจะศึกษาแบบออนไลน์ผ่านทาง Arduino Web Editor ก็ได้เช่นเดียวกัน ตัวอย่างที่ว่าประกอบด้วยรายการอุปกรณ์ (Hardware Required), การต่อวงจรบนเบรดบอร์ดหรือบอร์ดทดลอง (Breadboard Circuit), แผนผังวงจร (Schematic Diagram), ชุดคำสั่ง (Arduino Code) พร้อมคำอธิบายอย่างละเอียด ต่อจากนี้ผู้เขียนขอแสดงตัวอย่างคร่าวๆ เพื่อแสดงให้เห็นขั้นตอนการเรียนรู้ โดยจะขอข้ามคำอธิบายการทำงานของวงจรและโค้ดไปก่อน

สาธิตตัวอย่างการเรียนผ่าน Arduino Examples

Arduino Examples ประกอบด้วยตัวอย่างหลากหลายเพื่อให้เราได้ศึกษาเรียนรู้อย่างรอบด้าน เหมาะสำหรับผู้เริ่มต้นที่กำลังมองหาแนวทางการเรียนรู้ในภาคปฏิบัติ ควรใช้ที่นี่เป็นแหล่งศึกษาก่อนเป็นลำดับแรก เพื่อหาประสบการณ์กับการต่อวงจรอิเล็กทรอนิกส์ตามตัวอย่าง และเปิดตัวอย่างของโค้ดโปรแกรมขึ้นมาแล้วอัพโหลดลงบอร์ด เราก็จะเห็นผลลัพธ์ได้ทันที ซึ่งใน Examples มีบทเรียนให้ศึกษามากถึง 11 หัวข้อ แต่ละหัวข้อมีหลายตัวอย่างให้ทดลองใช้งานกัน

1. คลิกเมนู **File > Examples** จะปรากฏลิสต์ตัวอย่าง 11 หัวข้อ เราสามารถเลือกได้ตามความสนใจ (**Online** : https://create.arduino.cc, ให้ Create Account และ Log in)

Arduino IDE



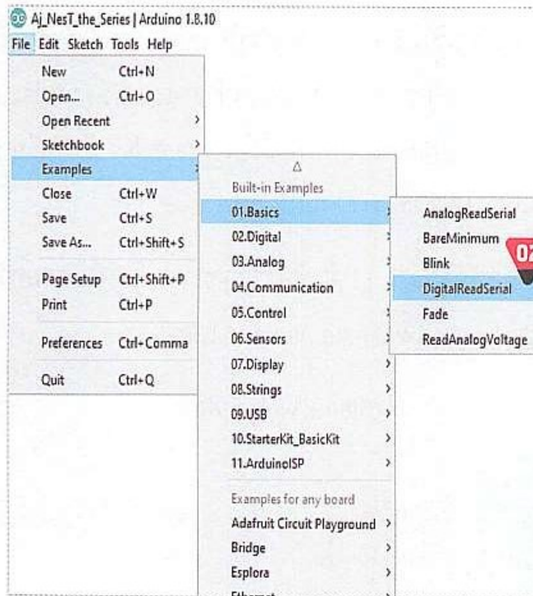
Arduino Web Editor



▲ รูปแสดงการเข้าถึงบทเรียนตัวอย่างในโปรแกรม Arduino IDE/Web Editor

2. ทดลองเปิดตัวอย่าง DigitalReadSerial ให้คลิกเลือก 01. Basics > DigitalReadSerial

Arduino IDE



Arduino Web Editor

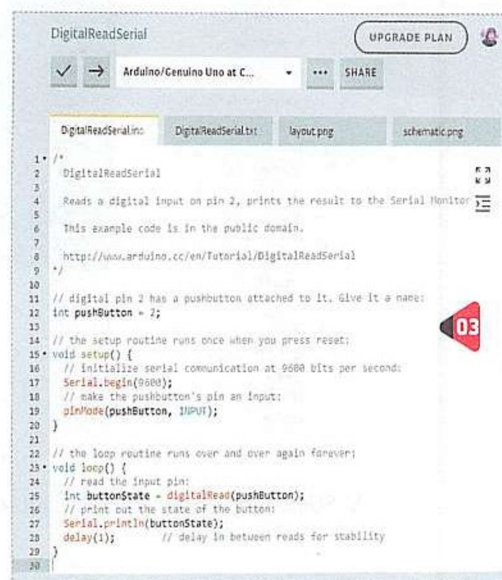


3. จะแสดงโค้ด DigitalReadSerial พร้อมคำอธิบายเกี่ยวกับการทำงานของโค้ด และมีลิงค์เชื่อมโยงไปยังเว็บเพจไปสู่เพจข้อมูลเพิ่มเติม

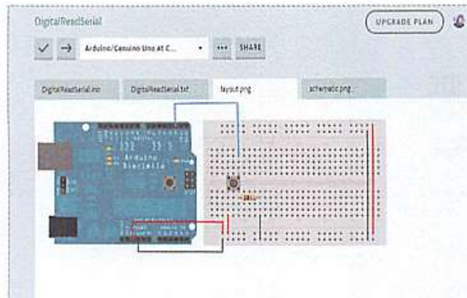
Arduino IDE



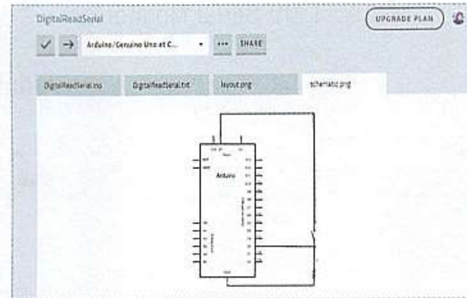
Arduino Web Editor



ใน Arduino Web Editor จะมีรูป Layout และ Schematic Diagrams ของวงจรให้ผู้เรียนได้ศึกษาการต่อวงจรไปด้วย

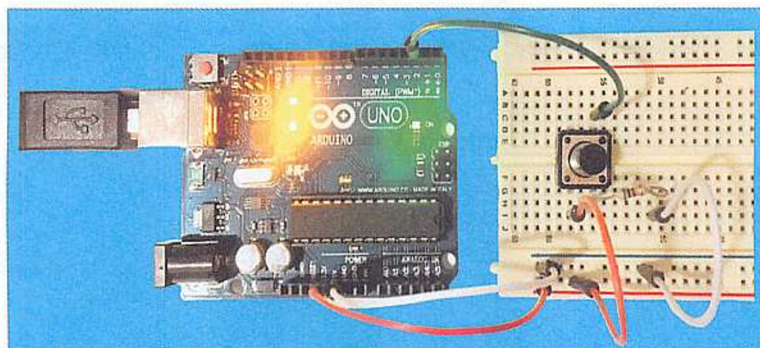


▲ รูปจำลองการต่อวงจรบน Breadboard



▲ Schematic แสดงแผนผังวงจร

4. ถ้ามีอุปกรณ์พร้อมก็ให้ต่อวงจรตามตัวอย่าง และต่อบอร์ด Arduino กับคอมพิวเตอร์ ดังรูป



5. ทำการ Verify และ Upload เพื่อป้อนโค้ด DigitalReadSerial ลงบอร์ด Arduino แล้วดูผลลัพธ์

Arduino IDE

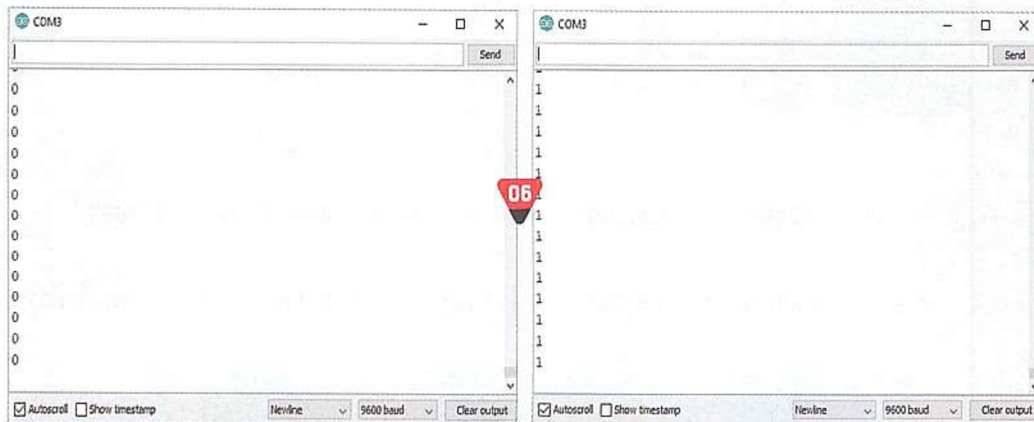


Arduino Web Editor

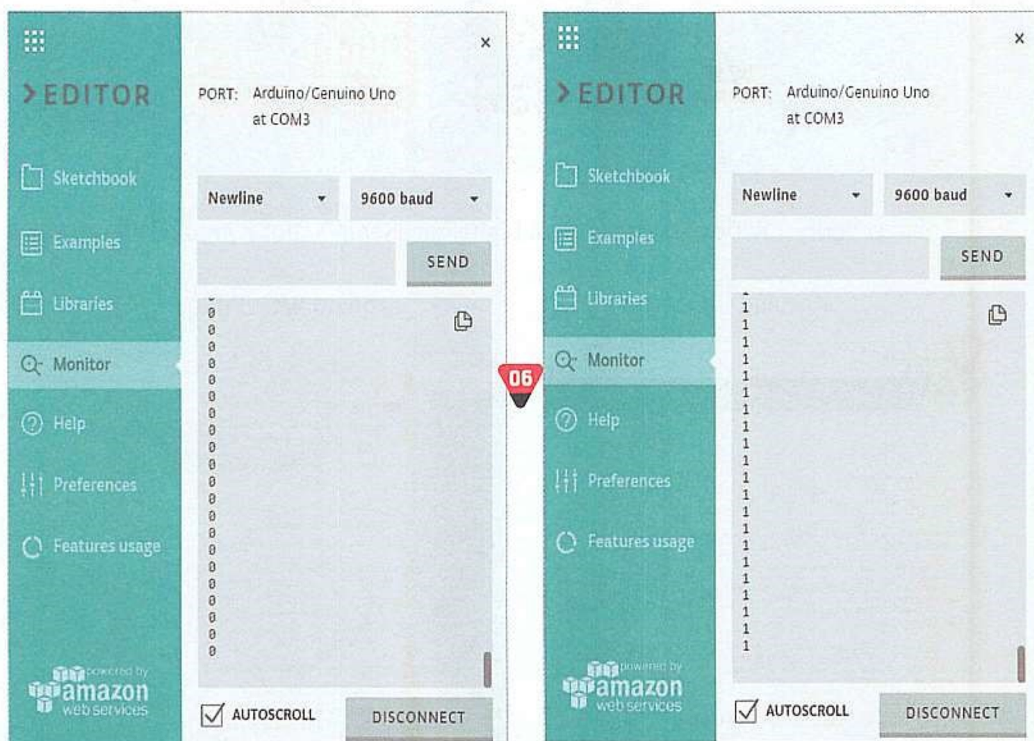


6. เปิด **Serial Monitor** ตั้งค่า **Baud Rate** ให้ตรงกับโค้ดเพื่อแสดงผลการทำงาน จะเห็นว่า เมื่อเรายังไม่ได้กดปุ่ม Switch จะแสดงสถานะเป็น 0 แต่เมื่อเรากดปุ่ม Switch จะแสดงสถานะตัวเลข 1 บน Serial Monitor

Arduino IDE



Arduino Web Editor





ศึกษารายละเอียดของตัวอย่างได้ที่ไหน

สังเกตในขั้นตอนที่ 3 จะมีลิงค์ในบรรทัดที่ 8 ซึ่งจะเชื่อมไปยังเว็บเพจ Document Project แสดงเอกสารของโปรเจ็กต์ตัวอย่าง ซึ่งในที่นี้คือ Digital Read Serial ประกอบด้วยหัวข้อ Hardware Required (ลิสต์อุปกรณ์), Circuit (การต่อวงจร), Schematic (แผนผังวงจร), Code (อธิบายการทำงานของโค้ด) และทั้งหมดนี้คือ ขั้นตอนที่เราจะใช้ฝึกฝนในภาคปฏิบัติช่วงเริ่มต้น หากสงสัยตรงไหนก็หาข้อมูลเพิ่มเติมได้ เช่น ไม่รู้จักอุปกรณ์บางตัวว่าทำหน้าที่อะไร วิธีต่อวงจรบนบอร์ดทดลอง การออกแบบวงจร Circuit และ Schematic หรือฝึกการเขียนภาษา C ด้วยตนเอง เป็นต้น

▲ รูปแสดงหน้าต่าง Arduino Web Editor ซึ่งจะมีลิงค์เชื่อมโยงไปยัง Document Project

สาธิตตัวอย่างการเรียนรู้จาก Arduino Library

Arduino Library คือ ที่เก็บรวบรวมชุดคำสั่งพร้อมใช้ เพิ่มความสะดวกในการใช้งานอุปกรณ์ต่อพ่วง เช่น Sensor, LCD Display, GPS, GSM, I2C, WIFI Shield, Modules และ Extension หรือส่วนต่อขยายต่างๆ ช่วยให้เราไม่ต้องเขียนโค้ดขึ้นเองทั้งหมด ทำเพียงแค่เพิ่ม Library ใน Arduino IDE หรือ Arduino Web Editor ก็สามารถนำมาเขียนโปรแกรมใช้งานได้ทันที ลดเวลาการเขียนโปรแกรมได้มาก เพราะไม่ต้องกังวลว่า จะเขียนโค้ดอย่างไรเพื่อใช้งานส่วนขยาย จะได้เอาเวลาไปสร้างสรรค์งานเพื่อให้เกิดประสิทธิภาพสูงสุดดีกว่า

จำนวน Arduino Library มีอยู่มากมาย ทั้งแบบ Standard Library ที่ติดตั้งมากับ Arduino IDE/ Web Editor หรือแบบที่ต้องดาวน์โหลดจากผู้พัฒนารายอื่นๆ

วิธีติดตั้ง Library ให้กับโปรแกรม Arduino

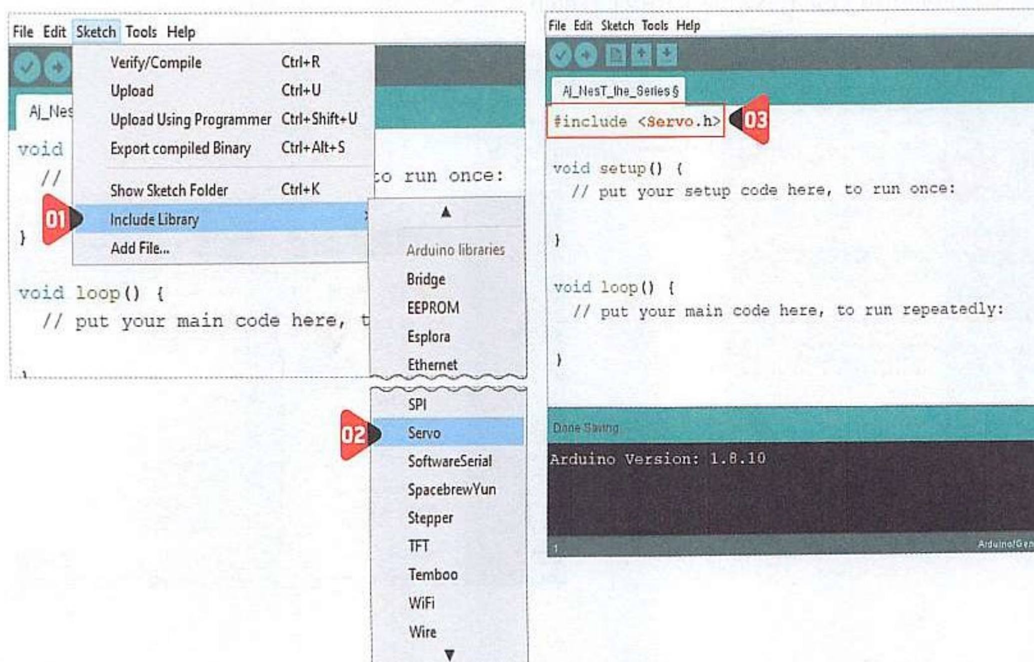
วิธีที่ 1 ติดตั้งโดยใช้ Built-in Library

ตัวอย่าง Library มาตรฐานของ Arduino ได้แก่

- **Bridge** : ช่วยให้การสื่อสารระหว่าง ATmega32U4 และ AR9331 ง่ายขึ้น
- **EEPROM** : สำหรับการอ่านและเขียนหน่วยความจำให้ EEPROM
- **Esplora** : สำหรับการเข้าถึงอุปกรณ์ Sensors และ Actuators ของบอร์ด Arduino Esplora
- **Ethernet** : สำหรับการเชื่อมต่ออินเทอร์เน็ตโดยใช้ร่วมกับ Arduino Ethernet Shield
- **Firmata** : สำหรับใช้สื่อสารกับโปรแกรมบนเครื่องคอมพิวเตอร์ด้วย Serial Protocol
- **GSM** : สำหรับการเชื่อมต่อกับเครือข่าย GSM/GPRS ด้วย GSM Shield
- **HID** : ช่วยให้บอร์ดรุ่น 32u4, Due และ Zero กลายเป็นอุปกรณ์ HID เพื่อรองรับ Mouse หรือ Keyboard หรืออุปกรณ์ HID ได้
- **Keyboard** : สำหรับเชื่อมต่อ Keyboard
- **LiquidCrystal** : สำหรับการควบคุมการแสดงผลหน้าจอ LCD
- **Mouse** : สำหรับเชื่อมต่อ Mouse
- **Robot Control** : สำหรับให้เข้าถึงฟังก์ชันการควบคุม Robot
- **Robot IR Remote** : สำหรับใช้ควบคุม Robot ด้วย Infrared Remote
- **Robot Motor** : สำหรับใช้ควบคุม Motor ของ Robot
- **SD** : สำหรับการอ่านและเขียน SD Card
- **SPI** : สำหรับใช้สื่อสารกับอุปกรณ์แบบ Serial Peripheral Interface
- **Servo** : สำหรับใช้ควบคุม Servo Motor
- **Stepper** : สำหรับใช้ควบคุม Stepper Motor
- **TFT** : สำหรับการแสดงผลตัวอักษร รูปภาพ และรูปทรงต่างๆ บนจอภาพ TFT
- **WiFi** : สำหรับการเชื่อมต่อกับอินเทอร์เน็ตโดยใช้ WiFi Shield

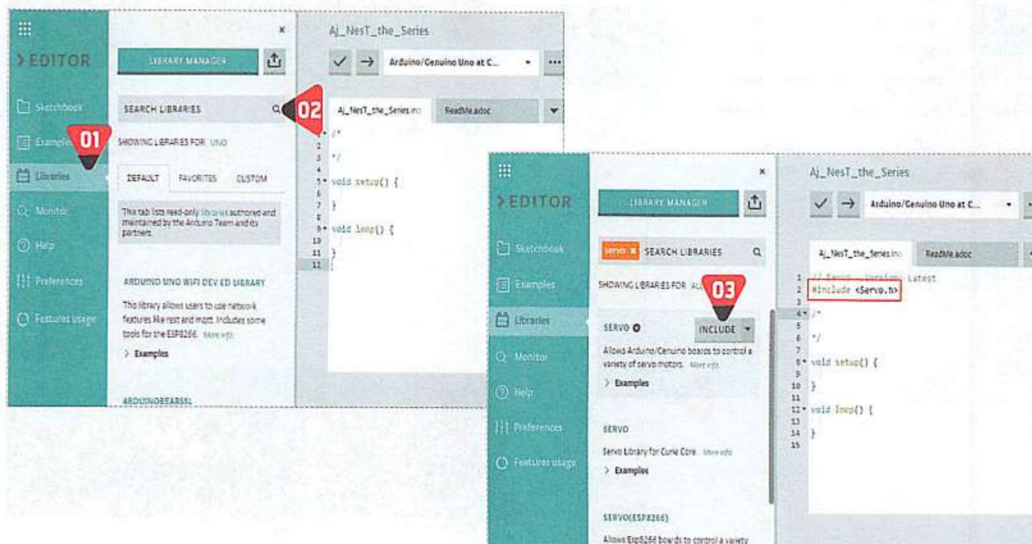
สำหรับโปรแกรม Arduino IDE

1. คลิกเมนู **Sketch > Include Library**
2. แล้วเลือก Library ที่ต้องการ ในที่นี้เลือก **Servo**
3. เมื่อเลือก Library ที่ต้องการแล้ว จะปรากฏคำว่า **#include <Servo.h>** เป็นอันติดตั้ง Library เสร็จเรียบร้อย



สำหรับโปรแกรม Arduino Web Editor

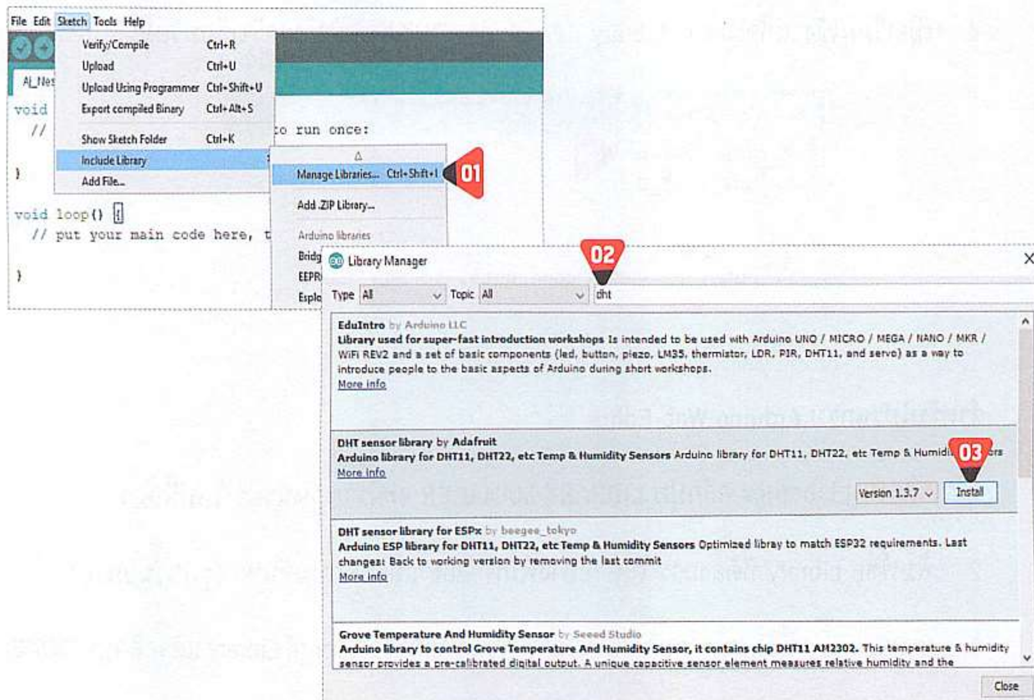
1. ไปที่หน้าเว็บเพจ **Arduino Web Editor** คลิกเมนู **Libraries**
2. ที่แท็บ **DEFAULT** พิมพ์ชื่อ Library ที่ต้องการค้นหาในช่อง **SEARCH LIBRARIES** ตัวอย่าง เช่น **servo** แล้วคลิกปุ่มค้นหา (รูปแว่นขยาย)
3. เมื่อผลการค้นหาปรากฏ ให้คลิกเลือกเวอร์ชันที่ต้องการ แล้วคลิกปุ่ม **INCLUDE** เพื่อติดตั้ง จากนั้น Library จะแสดงในหน้า Sketch



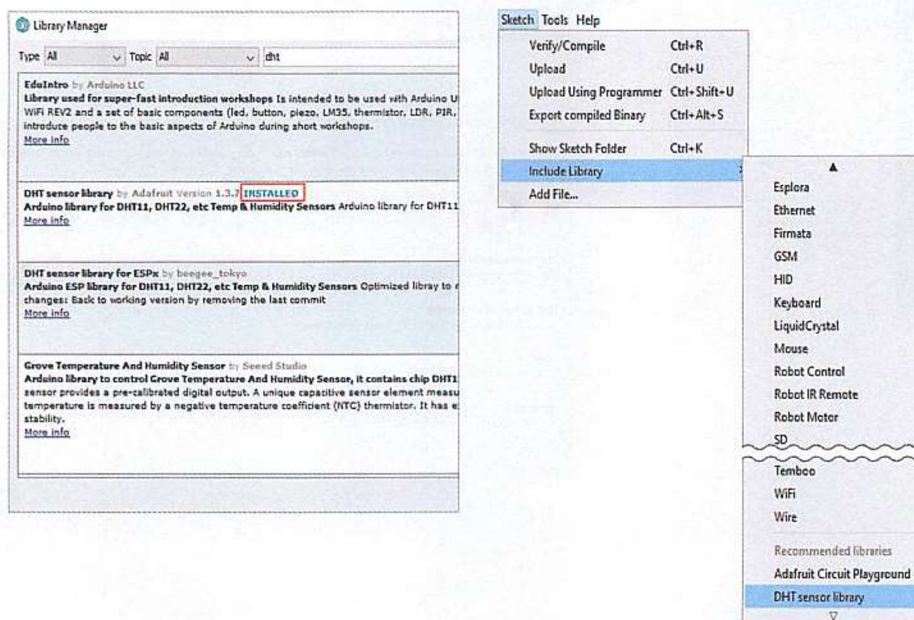
วิธีที่ 2 ติดตั้งจาก Library Manager

สำหรับโปรแกรม Arduino IDE

1. คลิกเมนู **Sketch > Include Library > Manage Libraries...**
2. พิมพ์ชื่อ **Library** ที่ต้องการในช่องค้นหา
3. เมื่อพิมพ์เสร็จจะแสดงลิสต์ของ Library ให้เลือก ให้คลิกเลือกรุ่นที่ต้องการแล้วคลิกปุ่ม **Install** สำหรับในตัวอย่างจะค้นหา **'dht'** ซึ่งใช้กับ DHT11 Sensor สำหรับตรวจจับอุณหภูมิ และความชื้น โดยเลือก **DHT sensor library by Adafruit** และทำการ **Install** (สามารถเลือก Type และ Topic ช่วยเพิ่มความสะดวกในการค้นหาได้)



เมื่อ Install เสร็จแล้วจะแสดงสถานะเป็น **INSTALLED** ต่อไปให้ตรวจสอบว่ามีการเพิ่ม Library ที่ติดตั้งใหม่แล้วหรือยัง คลิกเมนู **Sketch > Include Library** จะพบชื่อ **DHT sensor library** เพิ่มเข้ามาถูกต้อง



4. เมื่อเรียกใช้งานก็จะแสดง Library : #include <DHT.h> เป็นอันเสร็จเรียบร้อย

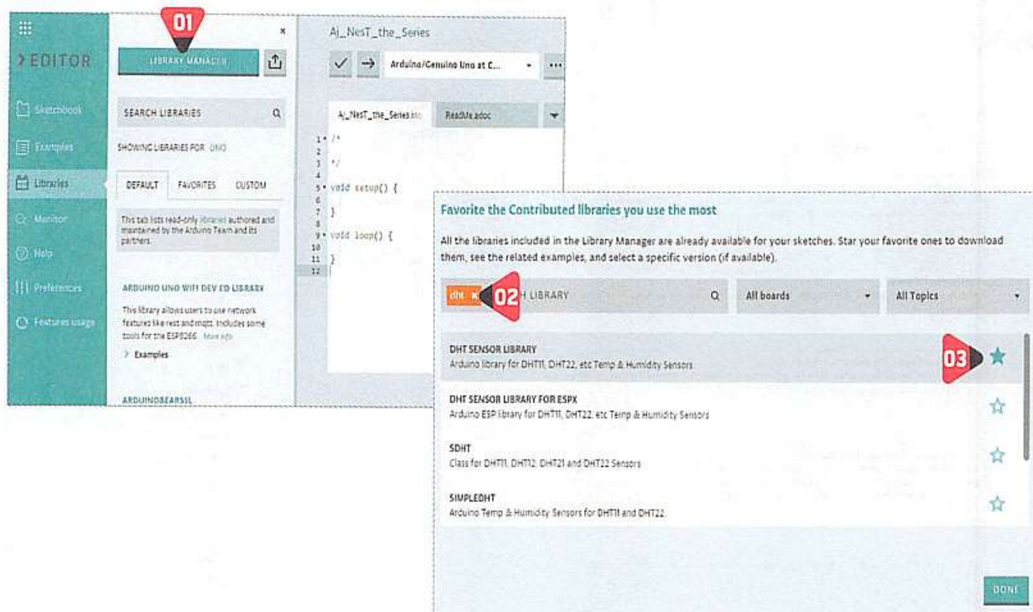
```
Aj_NesT_the_Series
#include <DHT.h>   04
#include <DHT_U.h>

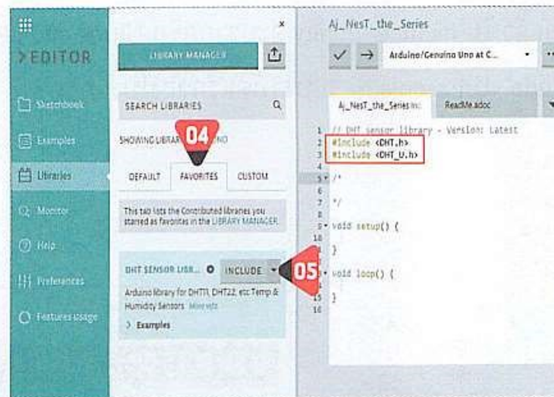
void setup() {
  // put your setup code here, to run once:

}
```

สำหรับโปรแกรม Arduino Web Editor

1. คลิกเมนู Libraries คลิกปุ่ม LIBRARY MANAGER จะปรากฏหน้าต่างใหม่ขึ้นมา
2. พิมพ์ชื่อ Library ที่ต้องการ ในตัวอย่างพิมพ์ 'dht' แล้วคลิกปุ่มค้นหา (รูปแว่นขยาย)
3. เลือก Library ที่ต้องการโดยคลิกที่รูปดาวเพื่อนำไปเก็บที่ Favorite Library แล้วคลิกปุ่ม DONE
4. คลิกแท็บ FAVORITES แล้วเลือก Library ที่ต้องการ
5. คลิกปุ่ม INCLUDE เพื่อติดตั้ง ก็จะพบ Library เพิ่มเข้าไปใน Sketch แล้ว



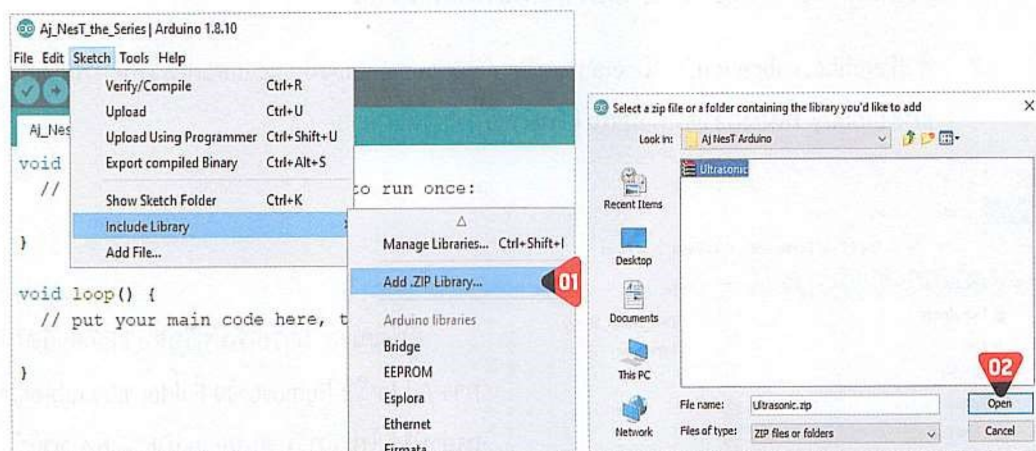


วิธีที่ 3 ติดตั้งเองผ่านไฟล์ที่ดาวน์โหลดมา

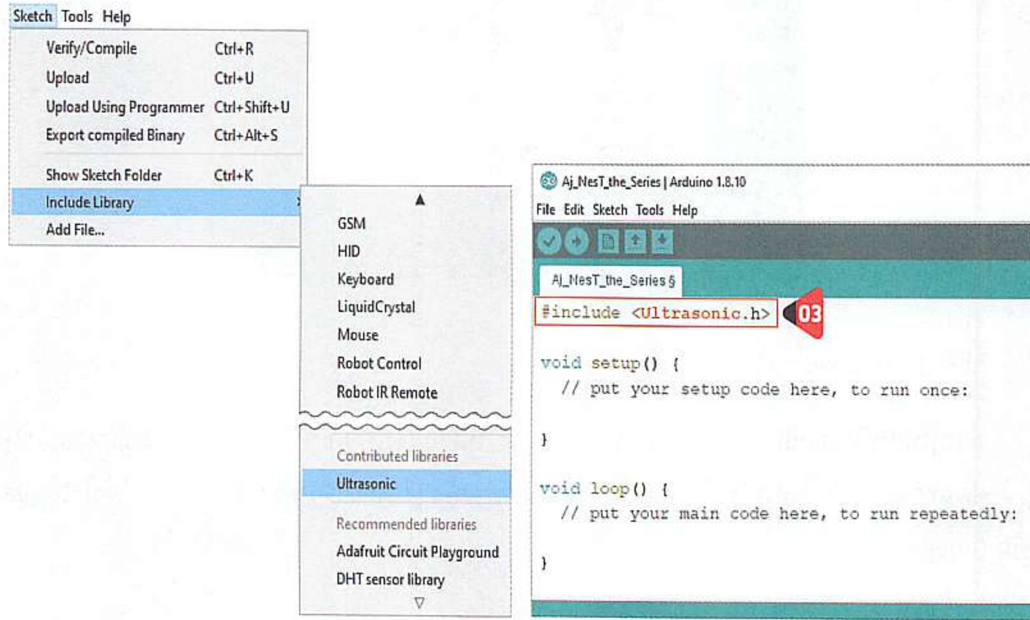
ส่วนใหญ่ใช้ในกรณีหาใน Library Manager แล้วไม่เจอ ซึ่งอาจเป็น Library ที่ใช้เฉพาะงาน วิธีนี้จะต้องค้นหาและดาวน์โหลดไฟล์ Library มาไว้ในเครื่องแล้ว (ไฟล์ .ZIP) ซึ่งสามารถค้นหาได้ที่ Google หรือ Github

สำหรับโปรแกรม Arduino IDE

1. คลิกเมนู **Sketch > Include Library > Add .ZIP Library...**
2. ไปยังไดเร็คทอรีที่เก็บไฟล์ แล้วเลือกไฟล์ Library ที่ต้องการติดตั้ง แล้วคลิกปุ่ม **Open** ในตัวอย่างใช้ไฟล์ Library ของ **Ultrasonic Sensor** ซึ่งเป็น Sensor สำหรับวัดระยะทางผ่านคลื่น Ultrasonic



3. เมื่อติดตั้งเสร็จแล้ว ใน Include Library จะปรากฏชื่อ Ultrasonic และเมื่อเลือก Ultrasonic ก็จะพบ **#include <Ultrasonic.h>** ในบรรทัดแรกของโปรแกรม

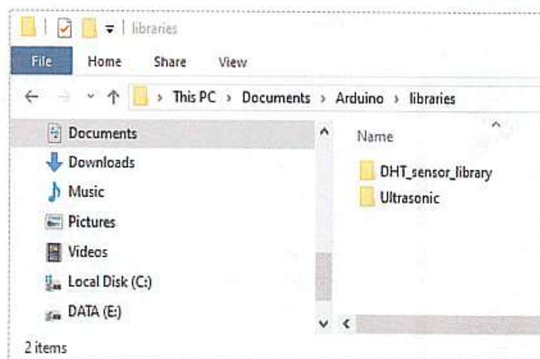


วิธีที่ 4 ติดตั้งโดยการแตกไฟล์ Library.ZIP

จะคล้ายกับวิธีที่ 3 เพียงแต่ต้องแตกไฟล์ .ZIP ก่อนแล้วนำไปใส่ในโฟลเดอร์ Library ของโปรแกรม

สำหรับโปรแกรม Arduino IDE

1. คลิกเมนู **File > Preferences** จะปรากฏหน้าต่างใหม่ขึ้นมา
2. ที่ **Sketchbook location: C:\Users\UserName\Documents\Arduino\libraries** ให้นำโฟลเดอร์ของ Library ที่ต้องการติดตั้งมาใส่ไว้ในไดเร็คทอรีดังกล่าว



ผู้พัฒนาสามารถเพิ่มหรือลบ Library โดยการ Add หรือ Remove ตัว Folder ของ Library เหล่านี้ได้ โปรแกรม Arduino IDE จะอัพเดตให้อัตโนมัติเมื่อเปิดโปรแกรมใหม่

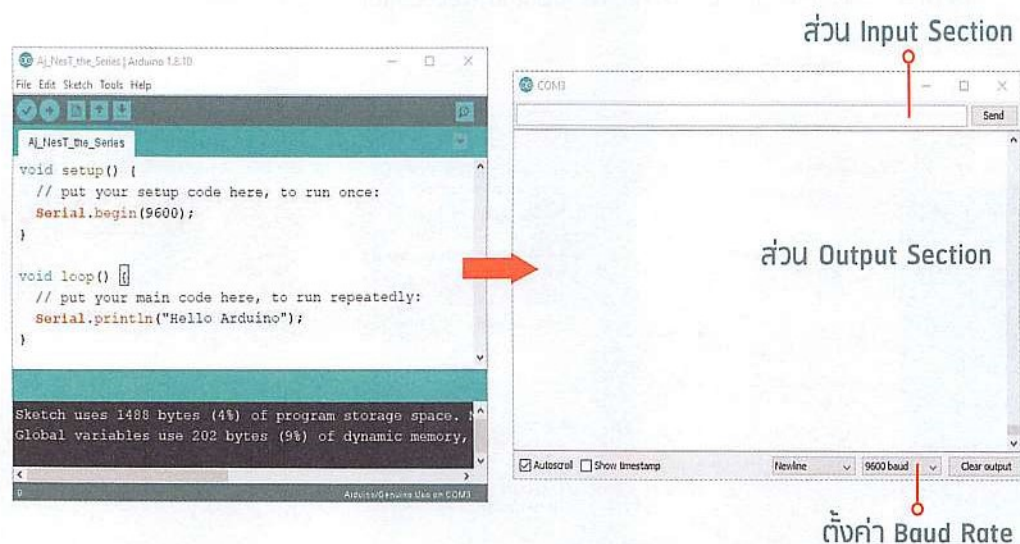
Serial Monitor

Serial Monitor เป็นฟีเจอร์ที่ใช้สำหรับรับส่งข้อมูล และแสดงผลข้อมูลผ่าน Serial Port ระหว่างบอร์ด Arduino และคอมพิวเตอร์ที่รันผ่านโปรแกรม Arduino IDE หรือ Arduino Web Editor ประกอบด้วย 2 ส่วน ได้แก่

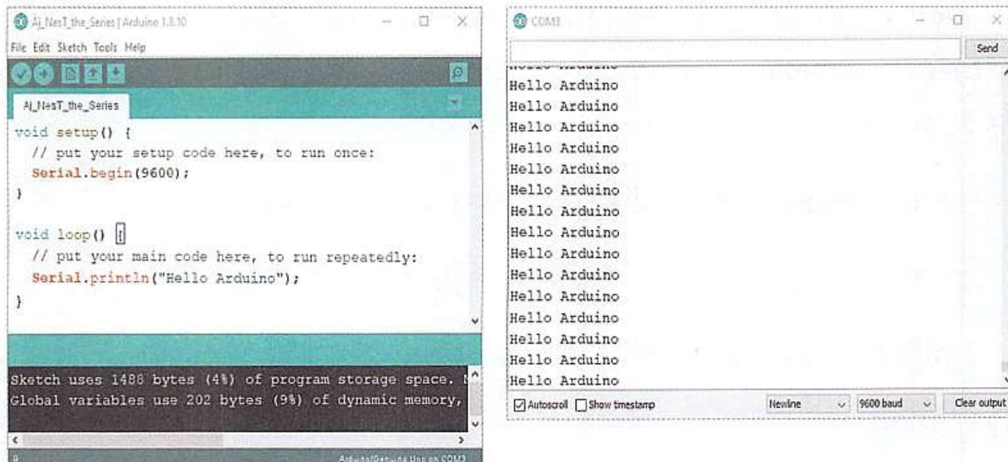
1. **Input Section** สำหรับป้อนข้อมูลเพื่อส่งให้ Arduino ประมวลผล
2. **Output Section** สำหรับแสดงผลลัพธ์ภายหลังการประมวลผลข้อมูลเสร็จแล้ว

การใช้งาน Serial Monitor จะต้องตั้งค่า Baud Rate ทั้งใน Sketch และใน Serial Monitor ให้ตรงกันจึงจะสามารถใช้งานได้

หน้าต่าง Serial Monitor ของโปรแกรม Arduino IDE

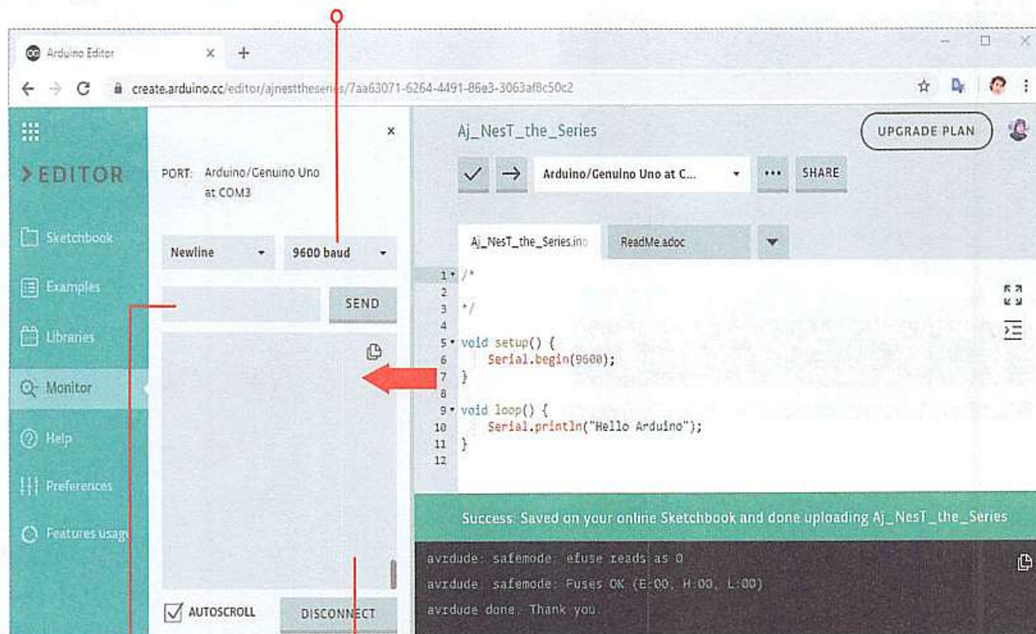


ทดสอบการแสดงผลข้อความ **"Hello Arduino"** บน Serial Monitor ในโปรแกรม Arduino IDE



หน้าต่าง Serial Monitor ของโปรแกรม Arduino Web Editor

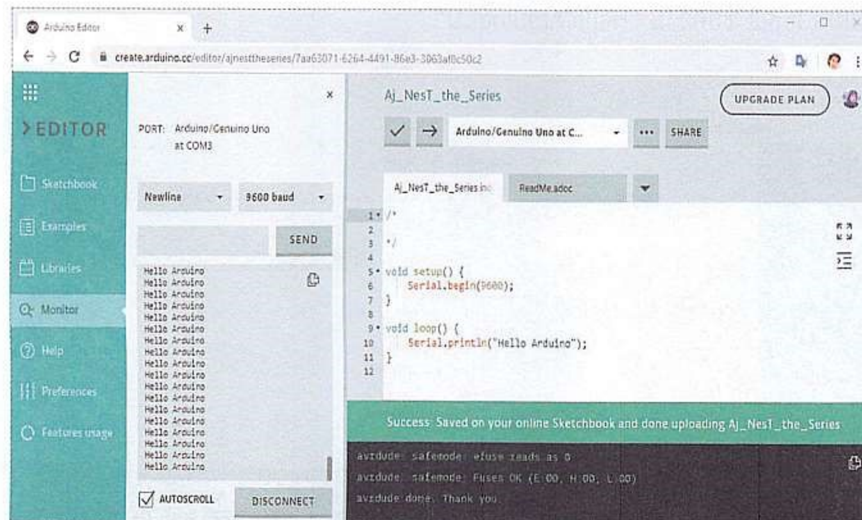
ตั้งค่า Baud Rate



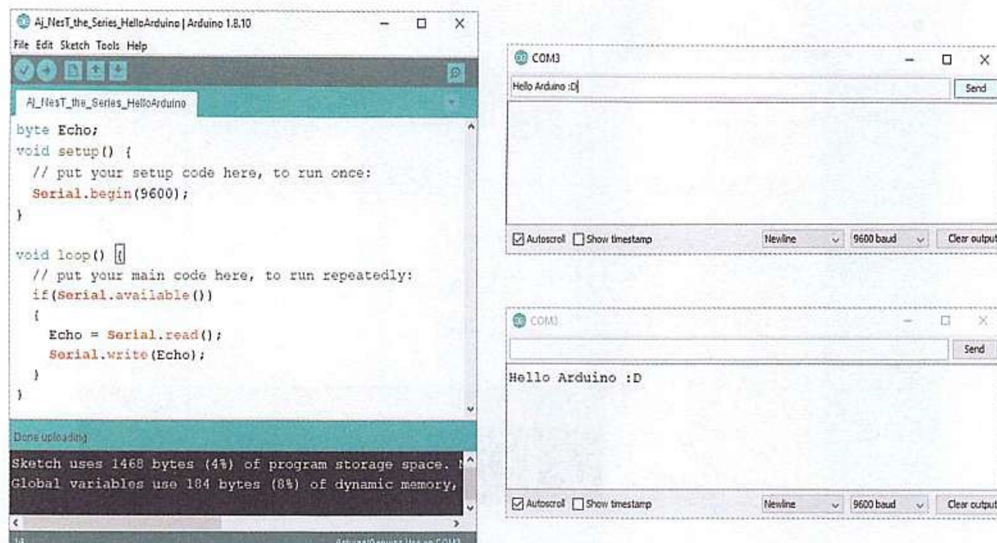
ส่วน Input Section

ส่วน Output Section

ทดสอบการแสดงผลข้อความ **"Hello Arduino"** บน Serial Monitor ในโปรแกรม Arduino Web Editor

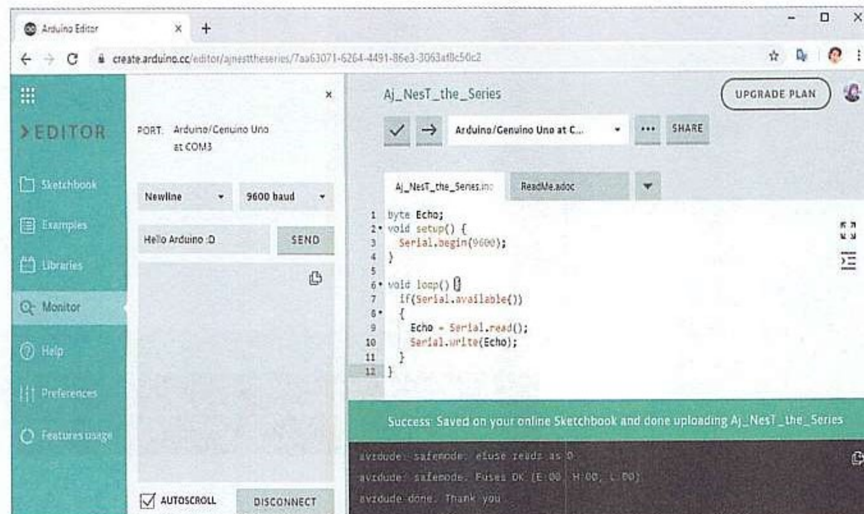


ทดสอบการรับข้อความจากการพิมพ์ **"Hello Arduino :D"** และแสดงออกบน Serial Monitor ในโปรแกรม Arduino IDE

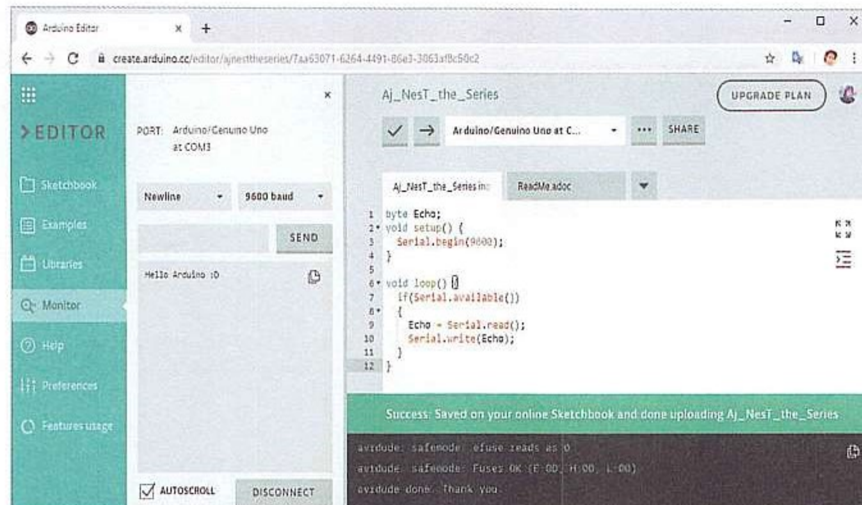


ทดสอบการรับข้อความจากการพิมพ์ **"Hello Arduino :D"** และแสดงออกบน Serial Monitor ในโปรแกรม Arduino Web Editor

รับข้อมูล Input ข้อความ "Hello Arduino :D"



แสดงข้อมูล Input ข้อความ "Hello Arduino :D" บนหน้าจอ Output ของ Serial Monitor





Arduino Create คืออะไร?

Arduino Web Editor หรือ **Coding Online** คือส่วนหนึ่งของ Arduino Create ซึ่งหมายถึงแพลตฟอร์มออนไลน์ที่เราสามารถใช้เพื่อเขียนโค้ด เข้าถึงบทเรียน ตั้งค่าให้กับบอร์ดที่ใช้งาน และแชร์โปรเจ็กต์ จะเห็นว่า มันถูกออกแบบมาเพื่อให้ผู้ใช้หรือนักพัฒนามีขั้นตอนการทำงานที่ต่อเนื่อง

Arduino Create คือ จุดเชื่อมโยงไปยังส่วนต่างๆ ตลอดเส้นทางของผู้พัฒนา เริ่มตั้งแต่สร้างแรงบันดาลใจไปจนถึงการนำไปปฏิบัติ ช่วยในการจัดการทุกสิ่งทุกอย่างในการทำโปรเจ็กต์ได้จากที่นี่ที่เดียว ฉะนั้น Arduino Create จึงเปรียบเหมือนกับศูนย์ควบคุมการทำงานที่รวมทุกอย่างไว้ในที่เดียว



▲ วิดีโอแสดงภาพรวมคร่าวๆ ของ **Arduino Online Editor** ที่มีอยู่ในแพลตฟอร์ม **Arduino Create**

YouTube : Search 'Arduino Create Editor', https://youtu.be/6cRFf4qkcTw

บทสรุปท้ายบท

ในบทนี้เราได้เรียนรู้วิธีการเริ่มต้นกับบอร์ด Arduino กันไปแล้ว รู้วิธีปรับแต่งค่าเพื่อให้ซอฟต์แวร์รู้จักกับฮาร์ดแวร์ และวิธีการศึกษาในภาคปฏิบัติที่เหมาะสมสำหรับผู้เริ่มต้นที่ต้องการ Guideline มีตัวอย่างมากมายที่ช่วยให้เราสามารถพัฒนาทักษะทางอิเล็กทรอนิกส์ และการเขียนโปรแกรม เชื่อว่าทุกคนจะได้รับประสบการณ์ที่ดีในการเรียนรู้บนแพลตฟอร์มออนไลน์ของ Arduino Cloud ในบทที่ 6 จะเน้นเรียนรู้การเขียนโปรแกรมควบคุมด้วยภาษา C และการต่อวงจรในภาคปฏิบัติทั้งต่อวงจรแบบจำลองและวงจรจริง

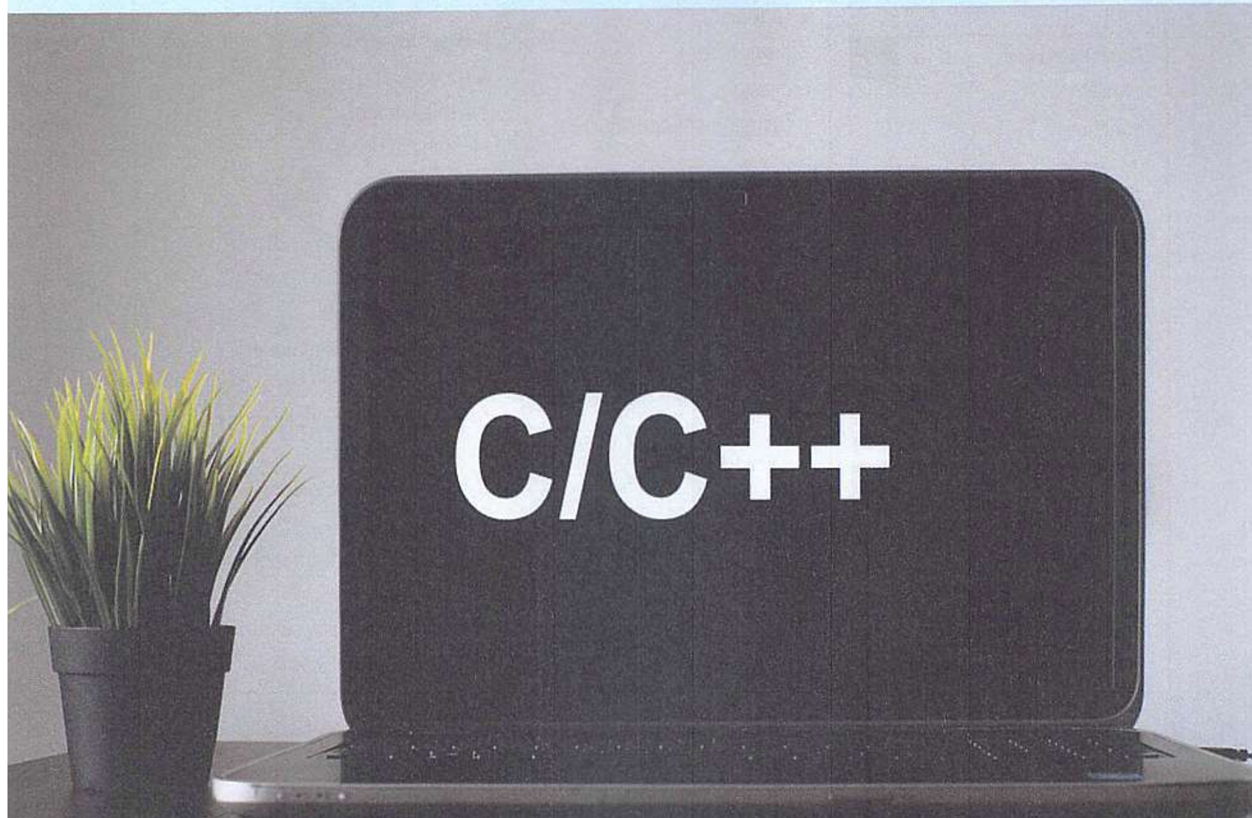
CHAPTER

06

การเขียนโปรแกรมควบคุมการทำงาน

Arduino Programming & Circuit Lab

ที่ผ่านมาเราได้เรียนรู้พื้นฐานการใช้งานซอฟต์แวร์ พร้อมทดลองเขียนโปรแกรมเบื้องต้น เพื่อเตรียมความพร้อมก่อนเริ่มเขียนโปรแกรมควบคุมการทำงานบอร์ด Arduino อย่างจริงจังกันในบทนี้ เริ่มตั้งแต่สรุปหลักการเขียนโปรแกรมด้วยภาษา C บนแพลตฟอร์ม Arduino พร้อมกันนี้ในหัวข้อ Arduino Tutorial จะเป็นการทำแล็บง่ายๆ ที่มีจุดประสงค์เพื่อให้ผู้เรียนเข้าถึงวิธีเขียนโปรแกรมในทางทฤษฎีและปฏิบัติไปพร้อมๆ กัน เพื่อสร้างความเข้าใจที่ลึกซึ้งมากยิ่งขึ้น



6.1 กลุ่มคำสั่งมีอะไรบ้าง (Arduino Programming)

เพื่อให้ทุกคนเข้าใจการเขียนโปรแกรม Arduino เราจะต้องรู้ก่อนว่ามีคำสั่งอะไรให้เราเล่นบ้าง หนังสือเล่มนี้ได้รวบรวมคำสั่งทั้งหมดโดยแบ่งออกเป็น 3 กลุ่ม ได้แก่ 1. Structures 2. Values และ 3. Functions

6.1.1 โครงสร้างทางไวยากรณ์ (Structures)

โครงสร้างทางไวยากรณ์ คือ กลุ่มคำสั่งประเภทโครงสร้างซึ่งในการเขียนโปรแกรมบน Arduino จะคล้ายกับการเขียนโปรแกรมโดยทั่วไป แต่ต่างกันตรงที่โครงสร้างการเขียนโปรแกรม Arduino จะมีฟังก์ชัน setup() และ loop() ในการทำงานเฉพาะของการเขียนโปรแกรมควบคุมบอร์ด Arduino

รายละเอียดคำสั่งประเภทโครงสร้าง ประกอบด้วย

1. **Sketch** : ตัวโปรแกรมหลัก
2. **Control Structure** : คำสั่งควบคุมการทำงาน
3. **Further Syntax** : ไวยากรณ์เสริม
4. **Operators** : ตัวดำเนินการต่างๆ

1 Sketch
loop()
setup()

2 Control Structure
break
continue
do...while
else
for
goto
if
return
switch...case
while

3 Further Syntax
#define (define)
#include (include)
/* */ (block comment)
// (single line comment)
; (semicolon)

4

Arithmetic Operators
% (remainder)
* (multiplication)
+ (addition)
- (subtraction)
/ (division)
= (assignment operator)

Comparison Operators
!= (not equal to)
< (less than)
<= (less than or equal to)
== (equal to)
> (greater than)
>= (greater than or equal to)

Boolean Operators
! (logical not)
&& (logical and)
|| (logical or)

Pointer Access Operators
& (reference operator)
* (dereference operator)

Bitwise Operators
& (bitwise and)
<< (bitshift left)
>> (bitshift right)
^ (bitwise xor)
| (bitwise or)
~ (bitwise not)

Compound Operators
%= (compound remainder)
&= (compound bitwise and)
*= (compound multiplication)
++ (increment)
+= (compound addition)
-- (decrement)
-= (compound subtraction)
/= (compound division)
^= (compound bitwise xor)
|= (compound bitwise or)

6.1.2 ตัวแปรและค่าคงที่ (Values ; Variables, Constants)

ตัวแปรและค่าคงที่ คือ กลุ่มคำสั่งประเภทกำหนดตัวแปร (Variables) และค่าคงที่ของตัวแปร (Constants) ของการเขียนโปรแกรม Arduino ในส่วนนี้จะคล้ายกับคำสั่งสำหรับการเขียนโปรแกรมโดยทั่วไป แต่ต่างกันตรงที่การกำหนดค่าคงที่ให้กับตัวแปรนั้น มีชื่อเฉพาะสำหรับเขียนโปรแกรมบน Arduino เช่น HIGH | LOW, INPUT | OUTPUT | INPUT_PULLUP, LED_BUILTIN เป็นต้น

รายละเอียดคำสั่งการจัดการค่าตัวแปรและค่าคงที่ ประกอบด้วย

1. Constants : การกำหนดค่าคงที่
2. Conversion : การแปลงประเภทข้อมูล
3. Data Types : ชนิดข้อมูล
4. Variable Scope & Qualifiers : การกำหนดขอบเขตและระบุชนิดข้อมูล
5. Utilities : คำสั่งช่วยอื่นๆ

Constants (1)
HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED_BUILTIN
true | false
Floating Point Constants
Integer Constants

Conversion (2)
(unsigned int)
(unsigned long)
byte()
char()
float()
int()
long()
word()

Data Types (3)
array
bool
boolean
byte
char
double
float
int
long
short
size_t
string
String()
unsigned char
unsigned int
unsigned long
void
word

Variable Scope & Qualifiers (4)
const
scope
static
volatile

Utilities (5)
PROGMEM
sizeof()

6.1.3 ฟังก์ชัน (Functions)

ฟังก์ชัน คือ กลุ่มคำสั่งประเภทฟังก์ชันที่ใช้เพื่อตั้งค่าการสื่อสาร และควบคุมการทำงานของการเขียนโปรแกรม Arduino ซึ่งจะต่างจากการเขียนโปรแกรมภาษาทั่วไปตรงที่มีฟังก์ชันที่ใช้สำหรับบอร์ด Arduino เช่น digitalRead(), digitalWrite(), pinMode() เป็นต้น

รายละเอียดคำสั่งประเภทฟังก์ชัน ประกอบด้วย

1. **Digital I/O** : ฟังก์ชันอ่าน เขียน และกำหนดขาพินอินพุต/เอาต์พุตแบบดิจิทัล
2. **Analog I/O** : ฟังก์ชันอ่าน อ้างอิง และเขียนแบบอะนาล็อก
3. **Zero, Due & MKR Family** : ฟังก์ชันอ่านและเขียนสำหรับบอร์ดตระกูล Zero, Due, MKR
4. **Advanced I/O** : ฟังก์ชันควบคุมอินพุต/เอาต์พุตขั้นสูง
5. **Time** : ฟังก์ชันควบคุมเวลา
6. **Math** : ฟังก์ชันด้านคณิตศาสตร์
7. **Trigonometry** : ฟังก์ชันตรีโกณมิติ
8. **Characters** : ฟังก์ชันด้านตัวอักขระ
9. **Random Numbers** : ฟังก์ชันสุ่มตัวเลข
10. **Bits and Bytes** : ฟังก์ชันควบคุมบิตและไบต์
11. **External Interrupts** : ฟังก์ชันอินเตอร์รัพต์ภายนอก
12. **Interrupts** : ฟังก์ชันอินเตอร์รัพต์ภายใน
13. **Communication** : ฟังก์ชันการสื่อสาร
14. **USB** : ฟังก์ชันสื่อสารกับอุปกรณ์ผ่านพอร์ต USB เช่น คีย์บอร์ดและเมาส์

Digital I/O (1)	Math (6)	Random Numbers (9)
digitalRead()	abs()	random()
digitalWrite()	constrain()	randomSeed()
pinMode()	map()	
	max()	
	min()	
	pow()	
	sq()	
	sqrt()	

Analog I/O (2)	Trigonometry (7)	Bits and Bytes (10)
analogRead()	cos()	bit()
analogReference()	sin()	bitClear()
analogWrite()	tan()	bitRead()
		bitSet()
		bitWrite()
		highByte()
		lowByte()

Zero, Due & MKR Family (3)
- analogReadResolution()
- analogWriteResolution()

Advanced I/O (4)
- noTone()
- pulseIn()
- pulseInLong()
- shiftIn()
- shiftOut()
- tone()

Time (5)
- delay()
- delayMicroseconds()
- micros()
- millis()

Characters (8)
- isAlpha()
- isAlphaNumeric()
- isAscii()
- isControl()
- isDigit()
- isGraph()
- isHexadecimalDigit()
- isLowerCase()
- isPrintable()
- isPunct()
- isSpace()
- isUpperCase()
- isWhitespace()

External Interrupts (11)
- attachInterrupt()
- detachInterrupt()

Interrupts (12)
- interrupts()
- noInterrupts()

Communication (13)
- Serial
- Stream

USB (14)
- Keyboard
- Mouse

จากการรวบรวมคำสั่งต่างๆ ของการเขียนโปรแกรมบน Arduino เพื่อให้เกิดความเข้าใจ เราจะมาเริ่มทำความรู้จักกับคำสั่งต่างๆ ผ่านการเขียนโปรแกรมว่ามีการทำงานและเรียกใช้งานอย่างไร และผู้อ่านสามารถศึกษารายละเอียดอื่นๆ เพิ่มเติมได้ที่เว็บไซต์ของ Arduino ตามลิงค์ต่อไปนี้

https://www.arduino.cc/reference/en/

6.2 วิธีเขียนโปรแกรมแบบ Arduino Sketch

การเขียนโปรแกรมบนแพลตฟอร์ม Arduino จะถูกเรียกว่า **"Sketch"** ซึ่งมีที่มาจากการสเกตซ์ภาพของศิลปินและนักออกแบบ ที่ใช้เพื่อร่างภาพในจินตนาการซึ่งเป็นวิธีที่ง่ายและรวดเร็วที่สุด และด้วยคำว่า "Sketch" ในความหมายดังกล่าว จึงได้ถูกนำมาใช้เรียกแทนตัวโปรแกรมที่เขียนขึ้นบน Arduino นั่นเอง

ภายในตัว Sketch จะแบ่งโค้ดออกแบบ 2 ส่วน ได้แก่ โค้ดส่วนที่ใช้รันเพียงครั้งเดียวเพื่อการตั้งค่าบอร์ดในฟังก์ชัน setup() และโค้ดส่วนที่รันในลักษณะวนซ้ำไปเรื่อยๆ ในฟังก์ชัน loop() ในทางปฏิบัติ คือ หลังจากโปรแกรมรันโค้ดส่วนของฟังก์ชัน setup() เสร็จแล้ว ก็จะเข้าสู่การรันโค้ดส่วนของฟังก์ชัน loop() ซึ่งจะเป็นการสื่อสารระหว่างบอร์ด Arduino กับอุปกรณ์ภายนอกต่างๆ เช่น เซนเซอร์, โมดูล หรือเครื่องคอมพิวเตอร์ ซึ่งเราจำเป็นจะต้องเขียนโค้ดคำสั่งเพื่อบอกกับบอร์ดว่า จะให้มันทำอะไรบ้างนั่นเอง

องค์ประกอบพื้นฐานในการเขียนโปรแกรมแบบ Arduino ที่จำเป็นต้องรู้ก่อนลงมือเขียนโปรแกรม Sketch ซึ่งจะช่วยให้ผู้เรียนมีความเข้าใจ และเกิดการเรียนรู้ที่รวดเร็ว มีดังนี้

6.2.1 โครงสร้างการเขียนโปรแกรม Arduino Sketch

สิ่งแรกที่ต้องรู้จัก Sketch คือ ตัวโปรแกรมที่เขียนบน Arduino Editor หรือ IDE นั้นจะประกอบด้วย 2 ฟังก์ชันหลัก คือ setup() และ loop() โดยตัวโค้ดที่แสดงให้เห็นนี้จะยังไม่ได้มีคำสั่งว่าต้องทำอะไร แต่อยากจะให้ดูโครงสร้างพื้นฐานของตัว Sketch ก่อน ซึ่งจะมีประโยชน์ตอนเขียนโค้ดเอง หรือนำโค้ดจากแหล่งอื่นๆ มาใส่ เพื่อจะได้รู้ว่าเราต้องเขียนหรือวางโค้ดตรงส่วนไหนนั่นเอง

```
/*
Arduino Tutorials
พื้นที่ประกาศตัวแปรครอบคลุมทุกฟังก์ชัน เช่น กำหนดขาสัญญาณที่ใช้ต่อกับบอร์ด Arduino
*/
void setup() {
 // ส่วนของ Setup Code จะรันคำสั่งเพียงครั้งเดียวหรือเมื่อมีการรีเซต (Reset)
}
void loop() {
 // ส่วนของ Main Code จะรันคำสั่งซ้ำๆ แบบวนลูป (Loop)
}
```

setup() [ฟังก์ชันบังคับที่ต้องมีทุกโปรแกรม] ฟังก์ชัน setup() จะถูกเรียกใช้เมื่อ Sketch เริ่มต้นการทำงานครั้งแรก ใช้สำหรับกำหนดค่าเริ่มต้นให้กับตัวแปร ใช้กำหนดหมายเลข PIN ทั้งในโหมด INPUT และ OUTPUT ใช้กำหนดค่านำเข้าการใช้งานไลบรารี (Library) หรือใช้เชื่อมต่อกับอุปกรณ์ภายนอกอื่นๆ โดยฟังก์ชันนี้จะทำงานเพียงครั้งเดียวหลังจากที่มีการเบิร์นโปรแกรมลงบอร์ด หรือเมื่อมีการกดปุ่มรีเซต (Reset) เพื่อเริ่มต้นการทำงานใหม่

loop() [ฟังก์ชันบังคับที่ต้องมีทุกโปรแกรม] หลังจากที่โปรแกรมรันส่วนของฟังก์ชัน setup() เพื่อเริ่มต้นตั้งค่าการทำงานเสร็จแล้ว โปรแกรมก็จะมารันส่วนของฟังก์ชัน loop() ต่อทันที โดยส่วนนี้จะเป็นการเขียนคำสั่งเพื่อควบคุมการทำงานของบอร์ด Arduino ให้ทำงานตามที่ต้องการในแบบซ้ำๆ ตามชื่อฟังก์ชันลูป โดยอาจจะเป็นการทำงานซ้ำๆ ตามจำนวนรอบที่กำหนด หรือทำซ้ำแบบไม่รู้จบก็ได้

6.2.2 การใส่คำอธิบายในโค้ด ; // และ /* ... */

องค์ประกอบที่สำคัญอีกส่วนหนึ่ง คือ การเขียน Comment หรือความคิดเห็นลงในโค้ด เพื่ออธิบายการทำงานด้วยภาษาที่เข้าใจง่าย ซึ่งการเขียน Comment ในโค้ดจะมีประโยชน์ทั้งกับตัวผู้เขียนโปรแกรมเองหรือผู้อื่นที่มาอ่านโค้ดเพื่อศึกษาการทำงานของโปรแกรม โดยข้อความ Comment นี้จะไม่ได้ถูกนำไปประมวลผลบนบอร์ด ตัวคอมไพเลอร์ (Compiler) จะซ่อนข้อความเหล่านี้เมื่อต้องการประมวลผล

การเขียน Comment สามารถเขียนได้ 2 วิธี ดังนี้

1. **การเขียน Comment บรรทัดเดียว** จะเขียนโดยใช้เครื่องหมาย // วางไว้ข้างหน้า Comment โดยจะวางหลังฟังก์ชันหรือคำสั่งในโปรแกรม เช่น // ตามด้วยข้อความ Comment
2. **การเขียน Comment หลายบรรทัด** จะเขียนโดยใช้เครื่องหมาย /* แล้วตามด้วยข้อความ Comment ปิดด้วยเครื่องหมาย */ (ไม่เกี่ยวกับกดปุ่ม <Enter>) ในโปรแกรม Arduino IDE จะปิด Comment โดยอัตโนมัติโดยเพิ่มคำสั่งปิด */ เช่น /* ข้อความ Comment หลายบรรทัด */

```
/*
  ข้อความ Comment หลายบรรทัด
*/
void setup() {
  Serial.begin(9600); // เปิดพอร์ตอนุกรมเพื่อเรียกใช้งาน Serial Monitor ที่ Baud Rate 9600
}
```

6.2.3 วงเล็บปีกกา { } และวงเล็บ ()

ภาษา Arduino มีฟังก์ชันเป็นเหมือนการกระทำหรือคำกริยาของภาษาโปรแกรม ซึ่งฟังก์ชันทั้งหมดจะเขียนตามด้วยเครื่องหมายวงเล็บเปิด (และวงเล็บปิด) ภายในวงเล็บจะมีการใส่ค่าข้อมูลอาร์กิวเมนต์ (Argument) หรือไม่มีก็ได้ และถ้าหากมีฟังชันที่ต้องการข้อมูล Argument หลายข้อมูล แต่ละ Argument จะต้องคั่นด้วยเครื่องหมายคอมม่า , เช่น void Setup(), Serial.begin(9600); หรือ Serial.begin(Add(2,3,4)); เป็นต้น ส่วนเครื่องหมายวงเล็บปีกกาเปิด { และปีกกาปิด } จะใช้แสดงขอบเขตเริ่มต้นและสิ้นสุดของฟังก์ชันหรือคำสั่งในโปรแกรม

```
void Setup()
{
  Serial.begin(9600);
  Serial.begin(Add(2,3,4));
}
```

6.2.4 เครื่องหมายเซมิโคลอน ;

เครื่องหมาย Semicolon ที่มีสัญลักษณ์ ; จะใส่ไว้ตรงท้ายคำสั่งต่างๆ เพื่อบอกให้คอมไพเลอร์รู้ว่าคำสั่งในบรรทัดนั้นเสร็จสมบูรณ์ หรือหากลืมใส่เครื่องหมาย ; จะส่งผลให้โปรแกรมเมื่อคอมไพล์เพื่อแปลงโค้ดไปเป็นภาษาเครื่อง จะมีการแจ้งเตือนข้อผิดพลาด (Error) เช่น Serial.begin(9600); Serial.println("Hello Arduino Programming"); เช่น

```
void setup()
{
  Serial.begin(9600); // ใส่ ; เพื่อบอกว่าคำสั่งเสร็จสมบูรณ์
  Serial.println("Hello Arduino Programming") // ถ้าไม่ใส่ ; จะเกิด Error แจ้งให้ทราบ
}
```

Arduino Tutorial 1 ทดลองเขียนข้อความบนจอ Serial Monitor

ทดสอบการแสดงผลข้อความออกบนจอ Serial Monitor โดยเขียนโปรแกรมแสดงผลข้อความ "Hello Arduino Programming" และ "I love Arduino Programming :D" ด้วยอัตราการส่งข้อมูลที่ 9600 bps ภายหลังจากจบการทำงานรอบแรกแล้ว โปรแกรมจะวนซ้ำการทำงานไปเรื่อยๆ ไม่รู้จบ

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

setup(): Serial.begin()
loop(): Serial.println, delay()

Syntax : Serial.begin(speed);

Serial.begin คือ คำสั่งตั้งค่าอัตราการส่งข้อมูลบิตต่อวินาที (Baud Rate) สำหรับการส่งข้อมูลแบบอนุกรมไปแสดงผลบนหน้าจอ Serial Monitor สามารถกำหนดอัตราการส่งข้อมูล Baud Rate ได้

Parameter : speed: ความเร็วของอัตราการส่งข้อมูล Baud Rate มีค่าตั้งแต่ 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, 38400, 57600 และ 115200 บิตต่อวินาที

Syntax : Serial.println(val);
Serial.print(val);

Serial.println() คือ คำสั่งแสดงผลข้อมูลเป็นข้อความไปยังพอร์ตอนุกรม Serial Monitor และขึ้นบรรทัดใหม่

Serial.print() คือ คำสั่งที่ใช้ในรูปแบบเดียวกันกับ Serial.println() แต่ต่างกันตรงที่ print จะไม่ขึ้นบรรทัดใหม่

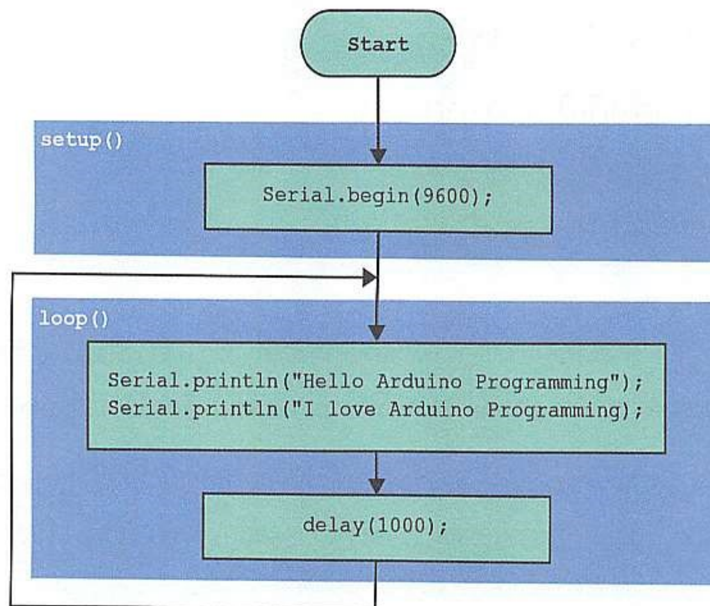
Parameter : val คือ ค่าที่ต้องการแสดง ถ้าเป็นข้อความจะใช้ "val" หากเป็นตัวเลขไม่ต้องใส่เครื่องหมาย " "

Syntax : delay(ms);

delay() คือ คำสั่งหน่วงเวลาการทำงานตามกำหนด แต่ยังคงแสดงผลตามคำสั่งก่อนหน้า สามารถตั้งเวลาได้ในระดับมิลลิวินาที เช่น delay(1000) หน่วงเวลาการทำงาน 1,000 มิลลิวินาที เท่ากับ 1 วินาที

Parameter : ms คือ ค่าเวลา หน่วยเป็นมิลลิวินาที (Millisecond)

Step 3 : Flowchart Arduino Tutorial 1



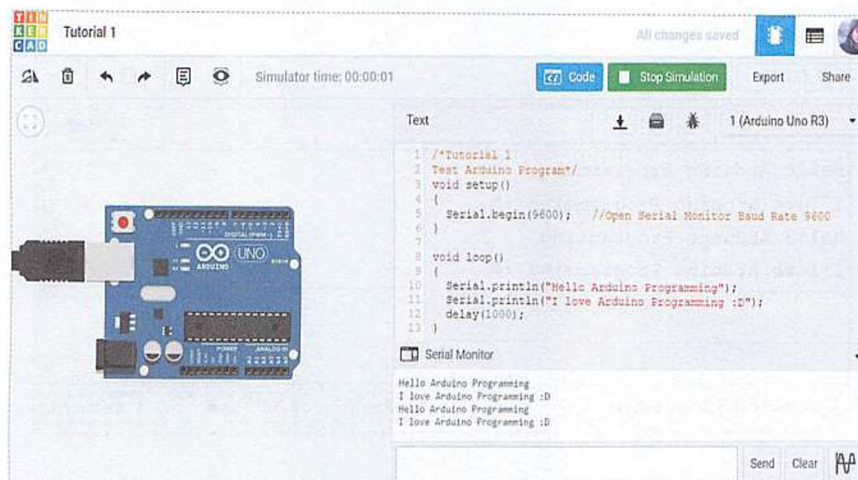
Step 4 : Source Code Arduino_Tutorial_1.ino

```
/*Tutorial 1
Test Arduino Program*/
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600); /          /เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //ฟังก์ชัน loop()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 Serial.println("Hello Arduino Programming");
                                //แสดงผลข้อความใน " " บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
 Serial.println("I love Arduino Programming :D");
                                //แสดงผลข้อความใน " " บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
 delay(1000);                   //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

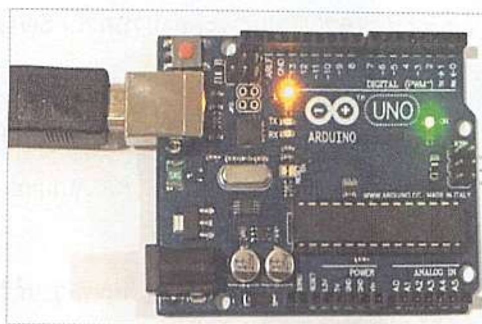
Step 5 : วิธีการทดลอง Arduino Tutorial 1

ให้ต่อวงจรและเขียนโปรแกรมบน Tinkercad ซึ่งเป็น Free Online Circuit Simulator แล้วรันดูผลลัพธ์ที่เกิดขึ้น เสร็จแล้วลองต่อวงจรจริงบนบอร์ด Arduino แล้วเขียนโปรแกรมบน IDE เพื่อรันดูผลลัพธ์เปรียบเทียบกัน

Tinkercad



Arduino Board

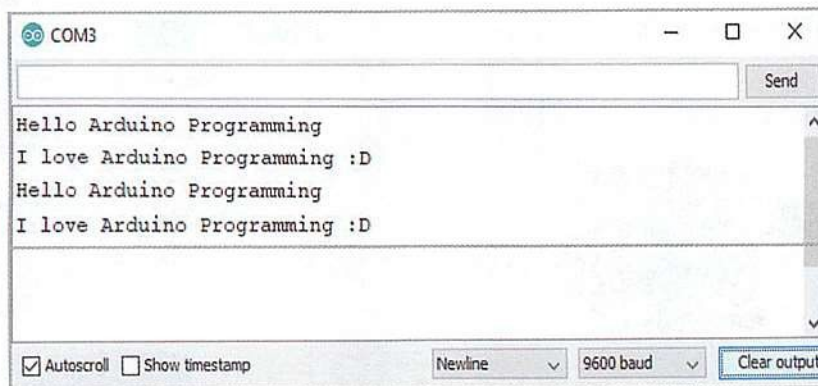



```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hello Arduino Programming");
  Serial.println("I love Arduino Programming :D");
  delay(1000);
}
```

Step 6 : ผลการทดลอง Arduino Tutorial 1

ใน Arduino Tutorial 1 นี้เราจะใช้ทั้งโปรแกรมออนไลน์ Tinkercad จำลองบอร์ด Arduino และใช้บอร์ดจริง Arduino ในการเขียนโปรแกรมแสดงผลลัพธ์บนหน้าจอ Serial Monitor โดยจะแสดงผลข้อความ "Hello Arduino Programming" แล้วขึ้นบรรทัดใหม่ และแสดงข้อความ "I love Arduino Programming :D" แล้วขึ้นบรรทัดใหม่ และแสดงผลข้อความทั้งสองวนซ้ำไปเรื่อยๆ ด้วยความเร็ว 1 วินาที



Step 7 : คำถามท้าย Arduino Tutorial 1

1. ถ้ากำหนดให้ Serial.begin(115200) แล้วจะต้องปรับค่าใน Serial Monitor อย่างไร เพื่อให้แสดงผลได้ถูกต้อง
2. ทดลองใช้คำสั่ง print แทนการใช้คำสั่ง println ผลลัพธ์ที่ได้เป็นอย่างไร
3. ทดลองเขียนโปรแกรมแสดงข้อความแนะนำตัว "สวัสดี Arduino ฉันชื่อ... (ชื่อ นามสกุล)... ยินดีที่ได้รู้จักนะ"
4. ถ้าต้องการให้โปรแกรมหน่วงเวลาการแสดงผลที่ 2 วินาที จะแก้ไขโปรแกรมอย่างไร

Arduino Tutorial 2 ทดลองเขียนโปรแกรมไฟกะพริบ

นอกจากการแสดงผลข้อมูลบนหน้าจอ Serial Monitor แล้ว Arduino ยังสามารถควบคุมการทำงานของอุปกรณ์ภายนอกอื่นๆ ได้อีกด้วย เช่น การสร้างไฟกะพริบด้วย Arduino โดยเราจะใช้ Syntax บางตัวที่ได้อธิบายไปก่อนหน้านี้เขียนโปรแกรมสร้างไฟกะพริบ โดยเมื่อไฟติดให้แสดงข้อความ "Turn ON" ออกทางหน้าจอ และเมื่อไฟดับให้แสดงข้อความ "Turn OFF" ออกทางหน้าจอ โดยไฟจะติดดับไปเรื่อยๆ ไม่รู้จบ ในแล็บนี้จะใช้ตัวต้านทานควบคุมการไหลของกระแสไฟ ถ้ามีค่าความต้านทานมากจะยอมให้กระแสไฟฟ้าไหลผ่านได้น้อย ในทางกลับกัน ถ้ามีค่าความต้านทานน้อยจะยอมให้กระแสไฟฟ้าไหลผ่านได้มาก เป็นการปรับแรงดันและกระแสไฟฟ้าให้เหมาะสม เพื่อไม่ให้เกิดความเสียหายกับอุปกรณ์นั่นเอง

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino	1	บอร์ด
2. หลอดไฟ LED	1	ดวง
3. ตัวต้านทาน 220 โอห์ม	1	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay()
```

รายละเอียดการใช้งานคำสั่งเพิ่มเติมจาก Arduino Tutorial 1

Syntax : pinMode(Pin, Mode);

pinMode() คือ คำสั่งที่มีไว้สำหรับกำหนดการทำงานของขา Pin ที่ต้องการใช้งานให้ทำงานในโหมดเป็น INPUT หรือ OUTPUT การทำงานเป็น INPUT เช่น การสั่งให้อ่านค่าสถานะสวิตช์ไฟว่าเปิดหรือปิดอยู่ การทำงานเป็น OUTPUT เช่น การสั่งจ่ายแรงดัน 5 โวลต์ (Volt ; เขียนย่อๆ ว่า V)

Parameter : Pin คือ หมายเลขขา Pin บนบอร์ด Arduino
Mode คือ โหมด เพื่อตั้งค่าขา Pin ให้ทำงานเป็น INPUT หรือ OUTPUT

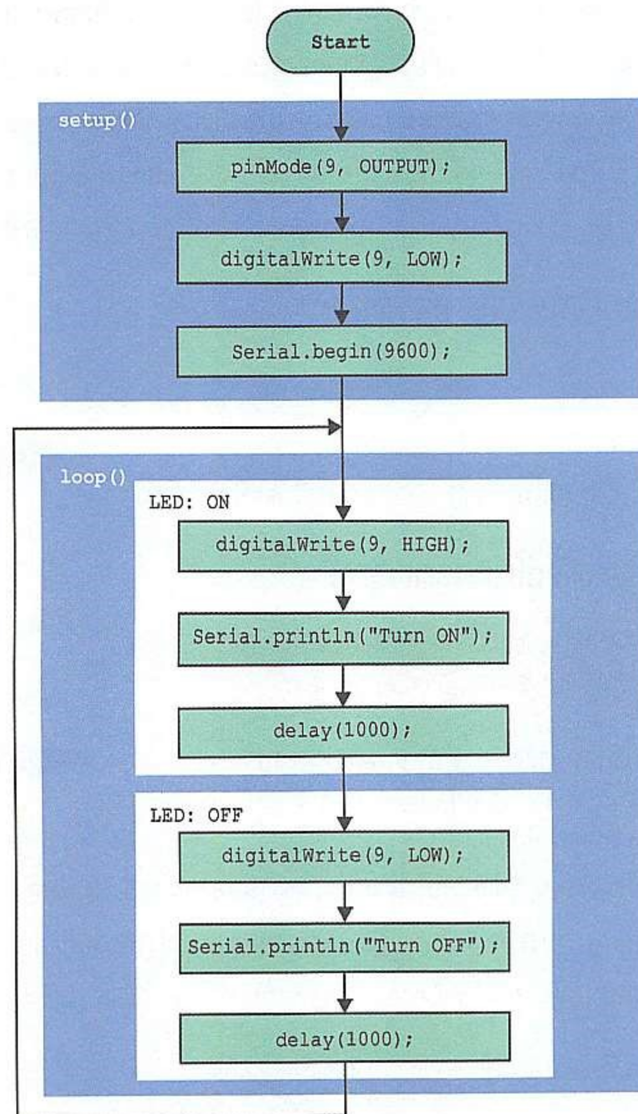
Syntax : digitalWrite (Pin, Value)

digitalWrite() คือ คำสั่งที่มีไว้กำหนดสถานะของขา Pin ที่ต้องการใช้งานให้มีค่าสถานะ 1 (HIGH = จ่ายแรงดัน 5 V) หรือ 0 (LOW = หยุดการจ่ายแรงดัน 5 V หรือลงกราวด์)

Parameter : Pin คือ หมายเลขขา Pin บนบอร์ด Arduino

Value : ค่า HIGH (เท่ากับ 1 V) หรือ LOW (เท่ากับ 0 V)

Step 3 : Flowchart Arduino Tutorial 2

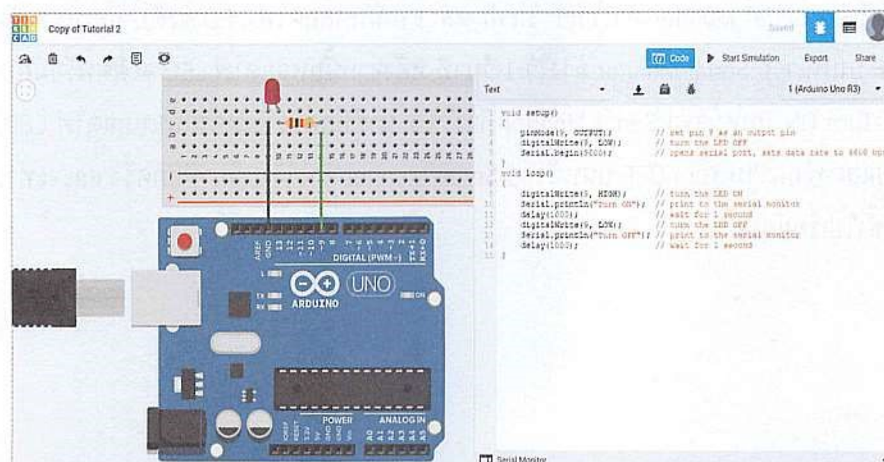


Step 4 : Source Code Arduino_Tutorial_2.ino

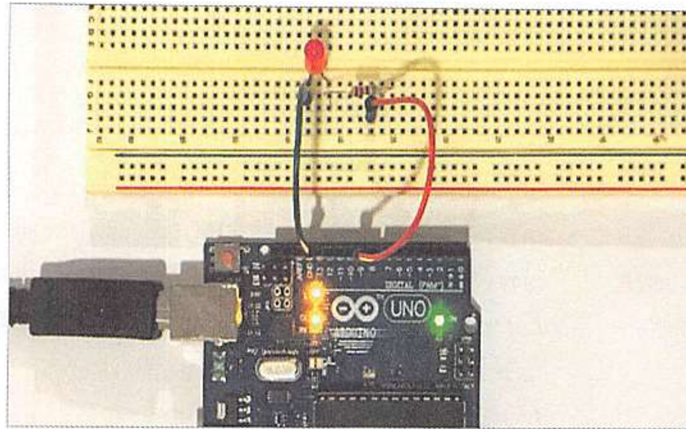
```
/*Tutorial 2
Blink a LED with Arduino*/
void setup()                 //ฟังก์ชัน setup()
{                            //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(9, OUTPUT);        //ตั้งขา Pin 9 ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(9, LOW);      //ตั้งขา Pin 9 ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);        //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                            //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                  //ฟังก์ชัน loop()
{                            //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  digitalWrite(9, HIGH);     //ตั้งขา Pin 9 ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println("Turn ON"); //แสดงผลข้อความ "Turn ON" บนจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);               //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(9, LOW);      //ตั้งขา Pin 9 ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println("Turn OFF"); //คำสั่งแสดงผลข้อความ "Turn OFF" บนจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);               //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                            //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 2

Tinkercad



Arduino Board

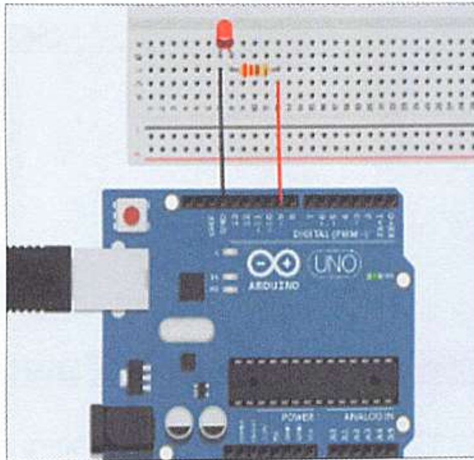


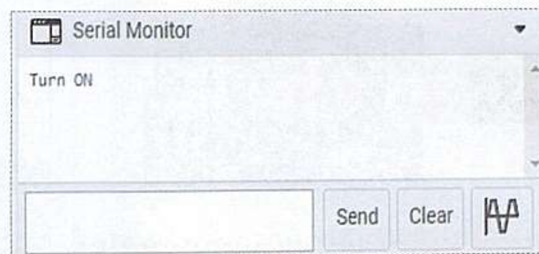
```
void setup()
{
    pinMode(9, OUTPUT);            // set pin 9 as an output pin
    digitalWrite(9, LOW);          // turn the LED OFF
    Serial.begin(9600);            // opens serial port, sets data rate to 9600 bps
}
void loop()
{
    digitalWrite(9, HIGH);         // turn the LED ON
    Serial.println("Turn ON");     // print to the serial monitor
    delay(1000);                   // wait for 1 second
    digitalWrite(9, LOW);          // turn the LED OFF
    Serial.println("Turn OFF");    // print to the serial monitor
    delay(1000);                   // wait for 1 second
}
```

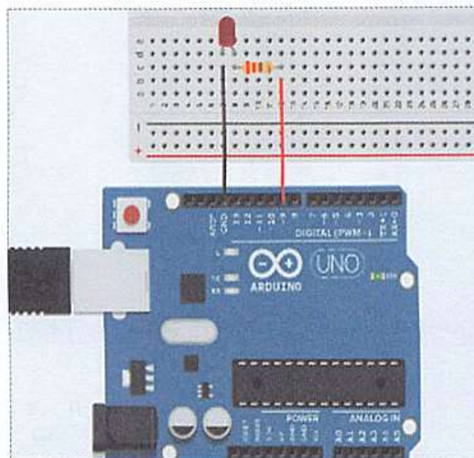
Step 6 : ผลการทดลองของ Arduino Tutorial 2

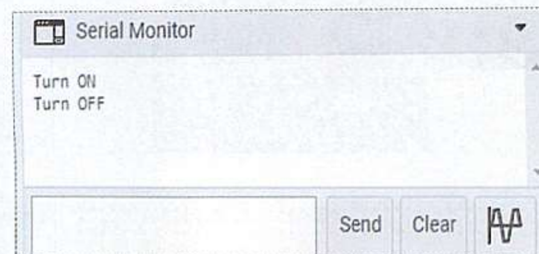
ผลลัพธ์เมื่อโหลดโปรแกรมลงบอร์ด Arduino หลอดไฟ LED จะติดพร้อมกับแสดงข้อความ Turn ON บนหน้าจอ Serial Monitor ค้างไว้ 1 วินาที หลังจากนั้นหลอดไฟ LED จะดับ และแสดงข้อความ Turn OFF บนหน้าจอ Serial Monitor ค้างไว้ 1 วินาที หลังจากนั้นหลอดไฟ LED จะติดพร้อมกับ แสดงข้อความ Turn ON บนหน้าจอ Serial Monitor ค้างไว้ 1 นาทีอีกครั้ง หลังจากนั้นหลอดไฟ LED จะดับพร้อมกับแสดงข้อความ Turn OFF บนหน้าจอ Serial Monitor ค้างไว้ 1 วินาทีอีกครั้ง และจะทำงานวนรอบเดิมซ้ำไปเรื่อยๆ

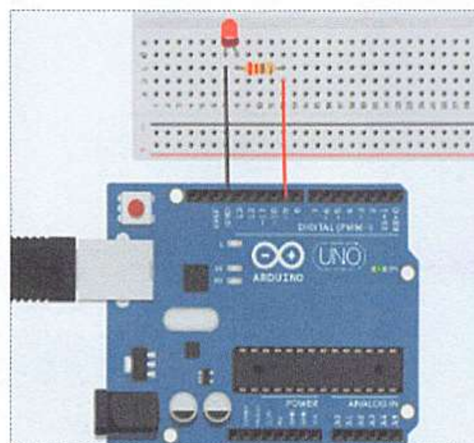
Tinkercad Output





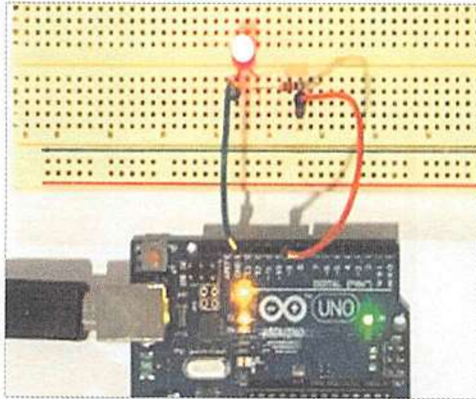


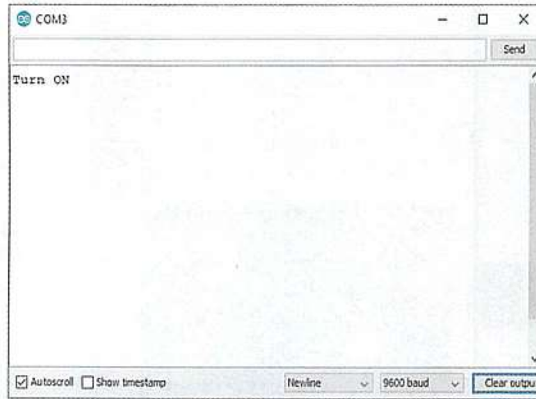


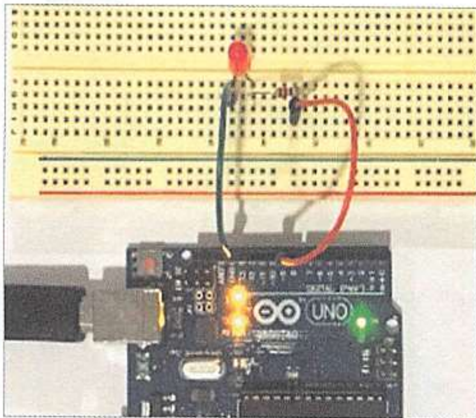


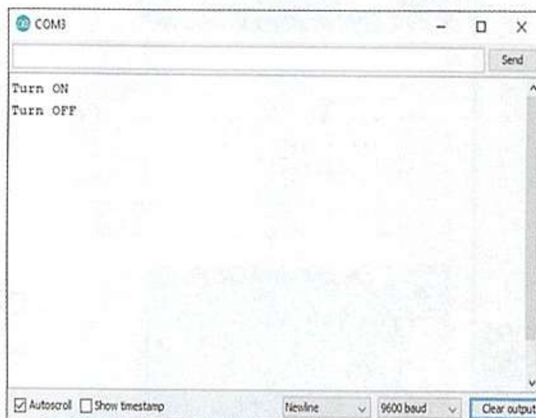


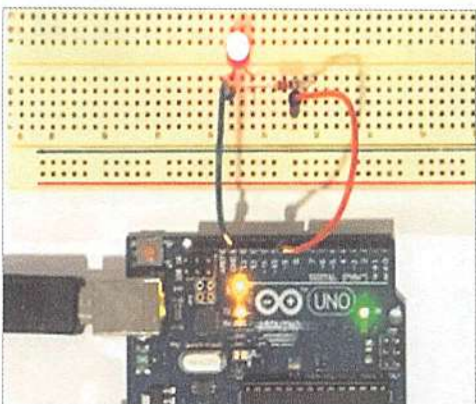
Arduino Board Output

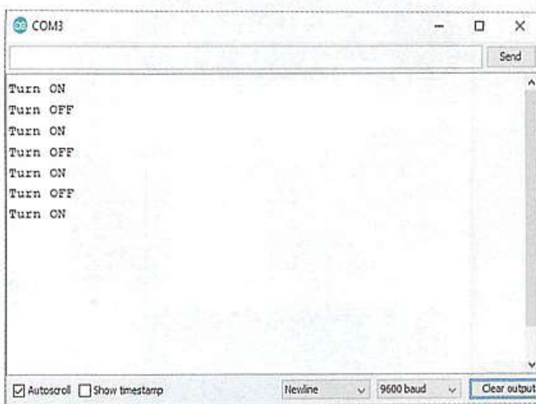












Step 7 : คำถามท้าย Arduino Tutorial 2

1. ทดลองแก้ไขโปรแกรมโดยเปลี่ยนหมายเลข Pin เป็น 13 จะต่อวงจรอย่างไร เพื่อให้รันผลลัพธ์ได้ถูกต้อง
2. ทดลองแก้ไขโปรแกรมโดยให้หลอดไฟ LED ติดค้างไว้ 3 วินาที และดับค้างไว้ 1 วินาที
3. ทดลองแก้ไขโปรแกรมโดยเมื่อหลอดไฟ LED ติด ให้แสดงข้อความ Status : Light ใต้ข้อความ Turn ON และเมื่อหลอดไฟ LED ดับ ให้แสดงข้อความ Status : Dark ใต้คำว่า Turn OFF ขึ้นบรรทัดใหม่
4. ทดลองแก้ไขโปรแกรมโดยให้เปลี่ยน LOW เป็นเลข 0 และเปลี่ยน HIGH เป็นเลข 1 ผลลัพธ์ที่ได้เป็นอย่างไร

6.3 การใช้งานตัวแปร (Arduino Variables)

ในการสร้างและกำหนดค่าตัวแปร โดยแบ่งออกเป็น 2 ประเภท ได้แก่ ตัวแปรภายนอกฟังก์ชัน (Global Variables) และตัวแปรภายในฟังก์ชัน (Local Variables) และเมื่อเราสร้างตัวแปรขึ้นมาจะถูกเรียกว่า “การประกาศ (Declaring)” ถ้าเราใส่ชื่อให้กับตัวแปรจะเรียกว่า “การตั้งชื่อ (Naming)”

Global Variable เป็นตัวแปรที่ประกาศอยู่ภายนอกฟังก์ชัน คือ ทุกฟังก์ชันในโปรแกรมสามารถมองเห็นได้ เช่น void setup(), loop(), void() เป็นต้น ส่วน **Local Variable** จะเป็นตัวแปรที่สามารถมองเห็นได้เฉพาะภายในฟังก์ชันที่ประกาศเท่านั้น เพราะเมื่อโปรแกรมเริ่มมีขนาดใหญ่และมีความซับซ้อนมากขึ้น การใช้ Local Variable จะเป็นวิธีที่มีประโยชน์ เพื่อให้แน่ใจว่ามีเพียงฟังก์ชันเดียวเท่านั้นที่สามารถเข้าถึงตัวแปรนี้ได้ เพื่อป้องกันโอกาสผิดพลาดในกรณีที่ฟังก์ชันหนึ่งแก้ไขตัวแปรที่ใช้โดยฟังก์ชันอื่น ก็จะไม่ส่งผลกระทบถึงกัน

6.3.1 ตัวแปรภายนอกฟังก์ชันและตัวแปรภายในฟังก์ชัน

ใน Arduino หากมีการประกาศตัวแปรที่ด้านบนของโปรแกรมก่อนฟังก์ชันการตั้งค่า void setup() ทุกส่วนของโปรแกรมจะสามารถใช้งานตัวแปรนี้ได้ หมายถึง ทุกฟังก์ชันจะเห็นตัวแปรนี้มีค่าตามที่ประกาศแบบครอบคลุมทุกฟังก์กัน ด้วยเหตุนี้ เราจึงเรียกตัวแปรภายนอกฟังก์ชันว่า “Global Variable” ในทางกลับกัน ถ้าเราประกาศตัวแปรระหว่างชุดของเครื่องหมายวงเล็บปีกกา { ... } ตัวแปรทุกตัวที่อยู่ภายใต้วงเล็บปีกกาเดียวกัน ก็จะใช้งานได้เฉพาะภายในพื้นที่วงเล็บปีกกาเท่านั้น คือ มองเห็นได้เฉพาะภายในฟังก์ชันเดียวกันเท่านั้น เราจึงเรียกตัวแปรภายในฟังก์ชันว่า “Local Variable”

ยกตัวอย่างเช่น ถ้าตัวแปรถูกประกาศอยู่ภายในฟังก์ชัน void setup() ตัวแปรนี้จะสามารถใช้งานได้เฉพาะภายใน void setup() เท่านั้น ไม่สามารถใช้งานในฟังก์ชัน void loop() ได้ เนื่องจากอยู่กันคนละฟังก์ชัน โดยมีวงเล็บปีกกาเป็นตัวกำหนดขอบเขตการมองเห็นตัวแปร ดังโค้ดตัวอย่าง

```
int x = 5;              // ทุกฟังก์ชันจะเห็นตัวแปร x มีค่าเท่ากับ 5
void setup()
{
 Serial.println(x);     // พิมพ์ค่าของตัวแปร x ออกทาง Serial Monitor
}
void loop()
{
 int i = 10;            // ค่าตัวแปร i จะเห็นได้เฉพาะภายในเครื่องหมาย {} ของฟังก์ชัน loop() เท่านั้น
}
```

6.3.2 การกำหนดค่าเริ่มต้นให้กับตัวแปร

ในการประกาศค่าตัวแปร เราจะเริ่มด้วยการ**ระบุชนิดข้อมูล (Data Type)** ตามด้วยชื่อตัวแปร ซึ่งในการเก็บค่าตัวแปรนั้นเราจะใช้**โอเปอเรเตอร์** หรือ**ตัวดำเนินการ (Assignment Operator)** เพื่อกำหนดค่าให้กับตัวแปร ซึ่งก็คือ เครื่องหมายเท่ากับ (=) แล้วปิดท้ายคำสั่งด้วยเครื่องหมาย Semicolon (;)

จากตัวอย่างที่ผ่านมา โปรแกรมเริ่มต้นบรรทัดแรกด้วยการประกาศค่าตัวแปร int x = 5;

แปรความคำสั่งนี้ได้ว่า int คือ การระบุชนิดข้อมูลให้กับตัวแปร (Data Types) ซึ่งในที่นี้คือ เลขจำนวนเต็ม (Integer Types) x คือ ชื่อตัวแปร เครื่องหมาย = คือ โอเปอเรเตอร์ และ 5 คือ ค่าที่เก็บในตัวแปร

สังเกตว่าเรามีการประกาศตัวแปรพร้อมใส่ค่าตัวแปรในบรรทัดเดียวกัน เราเรียกว่า "การกำหนดค่าเริ่มต้นให้กับตัวแปร (Initializing the Variable)"

รูปแบบของการกำหนดค่าตัวแปร DataType variable = value

ถึงแม้ว่าตัวแปร x จะเป็น Global Variable แต่เราก็ยังเปลี่ยนค่านี้ได้ในภายหลังภายในตัวโปรแกรม โดยการกำหนดค่าตัวแปรซ้ำอีกครั้ง เช่น int x = 10;

```
int x = 5;                   // ทุกฟังก์ชันจะเห็นตัวแปร x มีค่าเท่ากับ 5
void setup()
{
  Serial.begin(9600);        //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
  Serial.println(x);         // พิมพ์ค่าตัวแปร x ออกทาง Serial Monitor
}
void loop()
{
  int i = 10;                // ค่าตัวแปร i เห็นได้เฉพาะภายในเครื่องหมาย {} ของฟังก์ชัน loop() เท่านั้น
  int f = 5;                 // ค่าตัวแปร f เห็นได้เฉพาะภายในเครื่องหมาย {} ของฟังก์ชัน loop() เท่านั้น
       //...
  x = 10;                    // ตัวแปร x ขณะนี้มีค่าเท่ากับ 10
  Serial.println(x);         // พิมพ์ค่าตัวแปร x ออกทาง Serial Monitor
  delay(1000);               //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}
```

เมื่อรันโปรแกรมจะได้ผลลัพธ์ออกทาง Serial Monitor ดังนี้

```
5
10
10
...
10
```

Arduino Tutorial 3 ทดลองการใช้งานตัวแปร Global Variable และ Local Variable

เขียนโปรแกรมเพื่อเรียนรู้การใช้งานตัวแปรภายนอกฟังก์ชัน (Global Variable) และตัวแปรภายในฟังก์ชัน (Local Variable) โดยแสดงผลข้อมูลออกทางหน้าจอ Serial Monitor

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino	1	บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

Global Variable คือ ตัวแปร x ซึ่งอยู่นอกฟังก์ชันต่างๆ

setup(): Serial.begin(), Serial.println, delay()

loop(): Serial.println(), delay(), มี Local Variable คือ ตัวแปร x

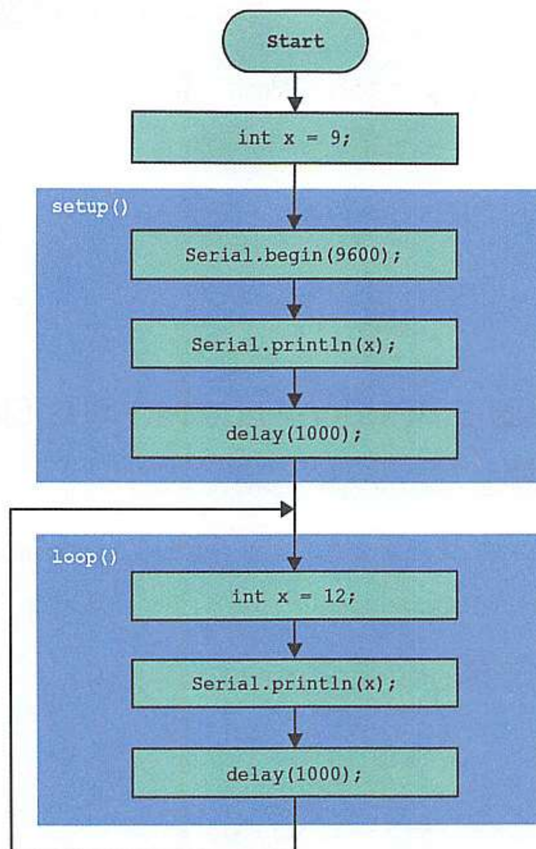
Syntax : DataType variable = value;

Parameter : DataType : ชนิดข้อมูลของตัวแปรที่สามารถเก็บได้ เช่น int, float, double, char, bool เป็นต้น ซึ่งจะกล่าวต่อไปในหัวข้อ Data Types

variable : ชื่อของตัวแปร ในตัวอย่างนี้คือ x

value : ค่าข้อมูลของตัวแปรที่ต้องการเก็บ

Step 3 : Flowchart Arduino Tutorial 3

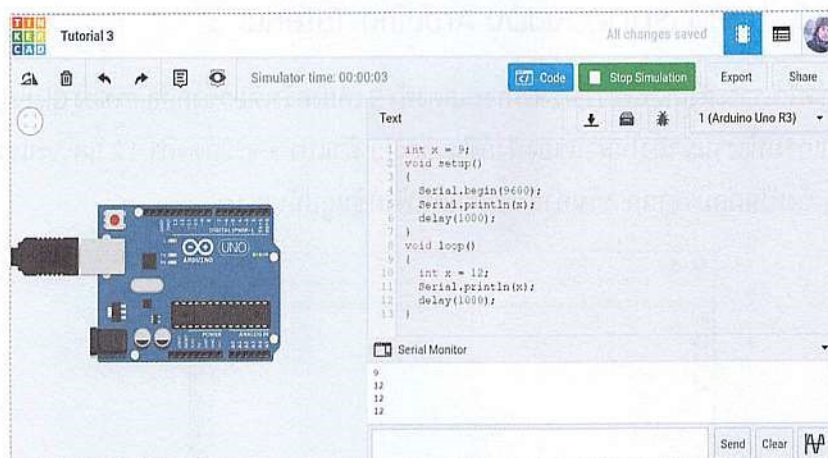


Step 4 : Source Code Arduino_Tutorial_3.ino

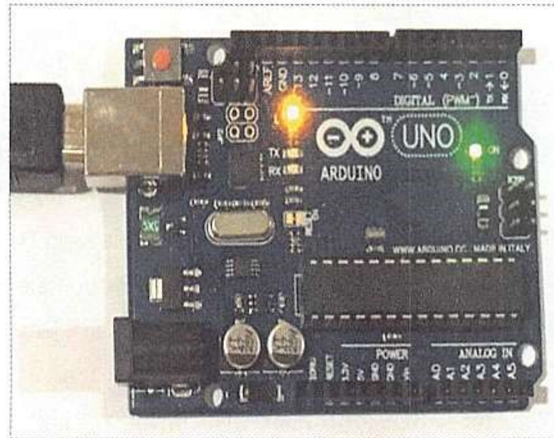
```
/*Tutorial 3
Global Variable and Local Variable*/
int x = 9;               //กำหนดค่าตัวแปร x ให้เป็น Global Variable เก็บตัวเลขจำนวนเต็มมีค่าเท่ากับ 9
void setup()             //ฟังก์ชัน setup()
{                        //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);     //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
 Serial.println(x);      //แสดงผลค่าข้อมูลของตัวแปร x บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
 delay(1000);            //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                        //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()              //ฟังก์ชัน loop()
{                        //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 int x = 12;             //กำหนดค่าตัวแปร x ให้เป็น Local Variable เก็บตัวเลขจำนวนเต็มมีค่าเท่ากับ 12
 Serial.println(x);      //แสดงผลค่าข้อมูลของตัวแปร x บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
 delay(1000);            //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                        //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 3

Tinkercad



Arduino Board

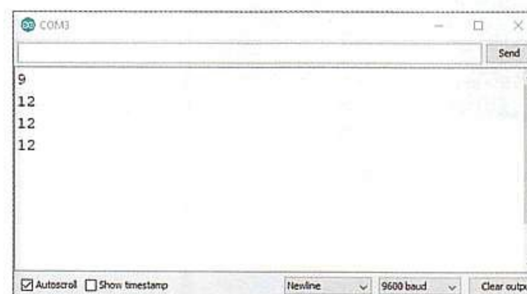


Arduino_Tutorial_3 §

```
int x = 9;
void setup()
{
  Serial.begin(9600);
  Serial.println(x);
  delay(1000);
}
void loop()
{
  int x = 12;
  Serial.println(x);
  delay(1000);
}
```

Step 6 : ผลการทดลองของ Arduino Tutorial 3

ผลลัพธ์ที่ได้เราจะสังเกตเห็นว่า ครั้งแรกจะแสดงค่า 9 เพียงครั้งเดียวซึ่งก็คือ ค่าของ Global Variable และเมื่อโปรแกรมทำงานมาถึงในส่วนของฟังก์ชัน loop() ตัวแปร x จะแสดงค่า 12 และจะแสดงค่า 12 ซ้ำไปเรื่อยๆ ซึ่งเป็นผลมาจากการรันคำสั่งซ้ำภายในฟังก์ชันลูปนั่นเอง



Step 7 : คำถามท้าย Arduino Tutorial 3

1. ถ้าต้องการให้ตัวแปร x ในฟังก์ชัน setup() แสดงค่าเป็นเลข 3 จะแก้ไขโปรแกรมอย่างไร
2. ถ้าใส่เครื่องหมาย // ตรงบรรทัดคำสั่ง int x = 12; ในฟังก์ชัน loop() ค่าของตัวแปร x ที่แสดงในฟังก์ชันนั้นเป็นเลขอะไร เพราะเหตุใด
3. ถ้ากำหนดให้ Global Variable x = 13 และสร้างตัวแปร Local Variable x = 6 ในฟังก์ชัน loop() ผลลัพธ์เป็นอย่างไร
4. ถ้าต้องการเพิ่มตัวแปร y เข้าไป โดยกำหนดให้ int y = 99 เป็น Global Variable และ int y = 13 เป็น Local Variable ในฟังก์ชัน loop() โดยให้มีการแสดงค่าข้อมูล y ออกทั้งฟังก์ชัน setup() และ loop() บน Serial Monitor จะต้องเขียนโปรแกรมอย่างไร

6.4 การตั้งชื่อตัวแปร (Naming the Variable)

การตั้งชื่อ (Naming) คือส่วนสำคัญอีกอย่างหนึ่งของตัวแปร แม้ว่าเราจะสามารถตั้งชื่ออะไรก็ได้ตามความต้องการ แต่ก็ควรมีข้อปฏิบัติที่ดีเพื่อให้เวลาเราอ่านโค้ดนี้ในภายหลัง จะยังคงเข้าใจว่า ตัวแปรตัวนี้คือตัวแปรอะไร ทำหน้าที่อะไร ซึ่งมีข้อควรปฏิบัติดังนี้

1. การตั้งชื่อตัวแปร ควรจะมีผลกับความเข้าใจในข้อมูลที่ถูกเก็บไว้ในตัวแปรนั้น เช่น การประกาศตัวแปรเพื่อตั้งค่าหมายเลขขา Pin ของหลอดไฟ LED เราอาจประกาศว่า int pinLed = 9; ถ้าลองตั้งชื่อใหม่ว่า pinMotor ก็จะเป็นการสื่อความหมายที่ผิด เพราะแทนที่จะเข้าใจว่าคือ pin ควบคุม LED กลับเข้าใจว่าคือ pin ควบคุมมอเตอร์
2. สังเกตว่าตัวอักษรตัวแรกในคำที่สองใน "pinLed" เป็นตัวพิมพ์ใหญ่ บางทีอาจจะดูว่าไม่จำเป็น แต่จะช่วยให้ผู้อ่านแยกแยะความแตกต่างระหว่าง 2 คำนี้ได้ง่าย
3. ตัวแปรอาจผสมด้วยตัวอักษรทั้งตัวพิมพ์ใหญ่ (A-Z) และตัวพิมพ์เล็ก (a-z)
4. ตัวแปรอาจผสมด้วยตัวเลข (0-9) แต่ไม่สามารถเริ่มต้นด้วยตัวเลขได้
5. ตัวแปรต้องไม่ใช้ชื่อเดียวกับคำสงวน (Reserved Word) หรือคำสั่งของการเขียนโปรแกรม Arduino
6. ตัวแปรต้องมีชื่อไม่ซ้ำกัน ยกเว้นตัวแปรประเภท Global Variable และ Local Variable สามารถใช้ซ้ำได้
7. ชื่อตัวแปรตัวพิมพ์ใหญ่และตัวพิมพ์เล็กจะไม่เหมือนกัน เช่น led กับ Led จะมองเป็นคนละตัวกัน
8. ชื่อตัวแปรจะต้องไม่มีอักขระพิเศษ ยกเว้นเครื่องหมายขีดเส้นใต้ _

Arduino Tutorial 4 ทดลองการตั้งชื่อตัวแปรที่ดี

จาก Arduino Tutorial 2 ในการเขียนโปรแกรม เราสามารถตั้งชื่อตัวแปรให้กับหมายเลข pin ของบอร์ด Arduino ได้

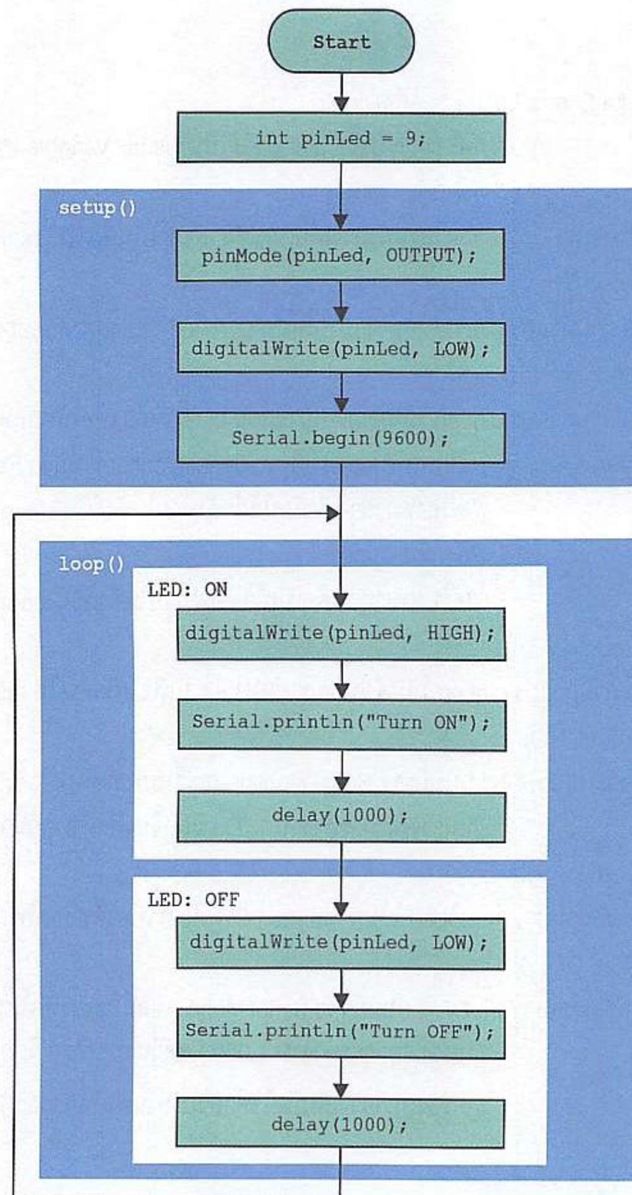
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino 1 บอร์ด
2. หลอดไฟ LED 1 ดวง
3. ตัวต้านทาน 220 โอห์ม 1 ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 4

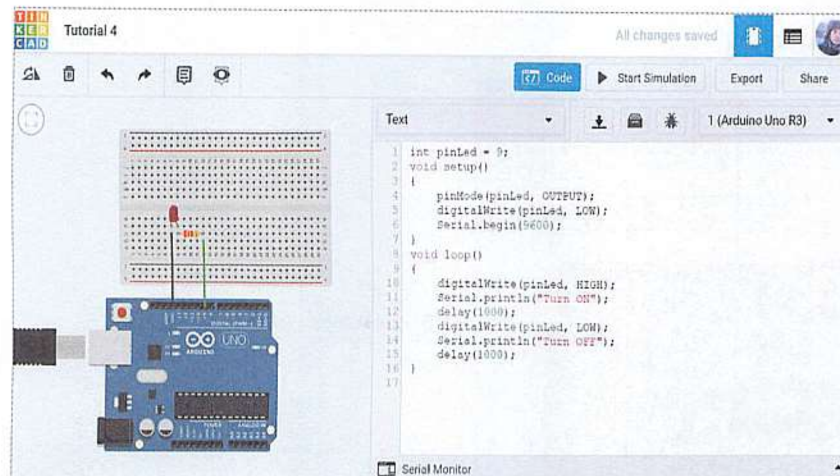


Step 4 : Source Code Arduino_Tutorial_4.ino

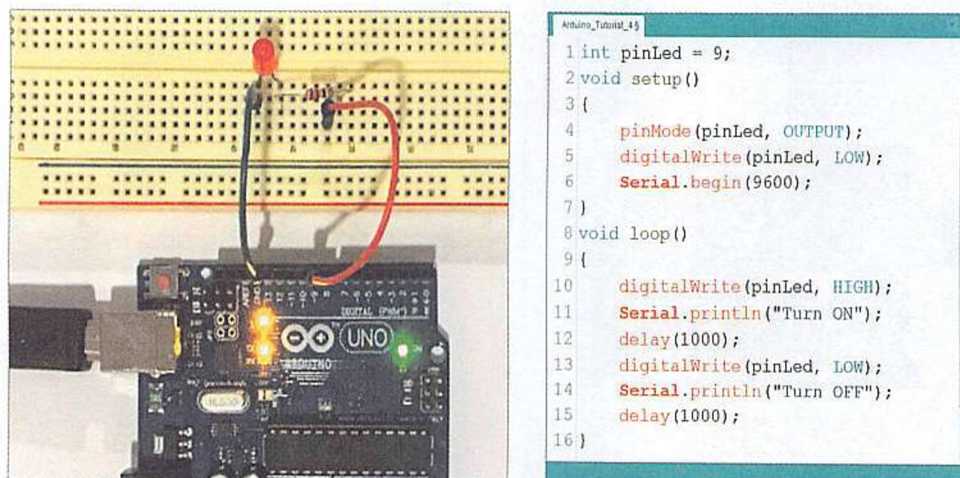
```
/*Tutorial 4
Naming Variables for Blink a LED with Arduino*/
int pinLed = 9;                //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable โดยใช้ Pin 9
void setup()                   //ฟังก์ชัน setup()
{                              //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(pinLed, OUTPUT);
    //ตั้งค่าขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(pinLed, LOW);
    //ตั้งค่าขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);          //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                              //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                    //ฟังก์ชัน loop()
{                              //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  digitalWrite(pinLed, HIGH);
    //ตั้งค่าขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println("Turn ON");
    //แสดงผลข้อความ "Turn ON" บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);                 //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
    //ตั้งค่าให้ขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println("Turn OFF");
    //คำสั่งแสดงผลข้อความ "Turn OFF" บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);                 //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                              //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 4

Tinkercad



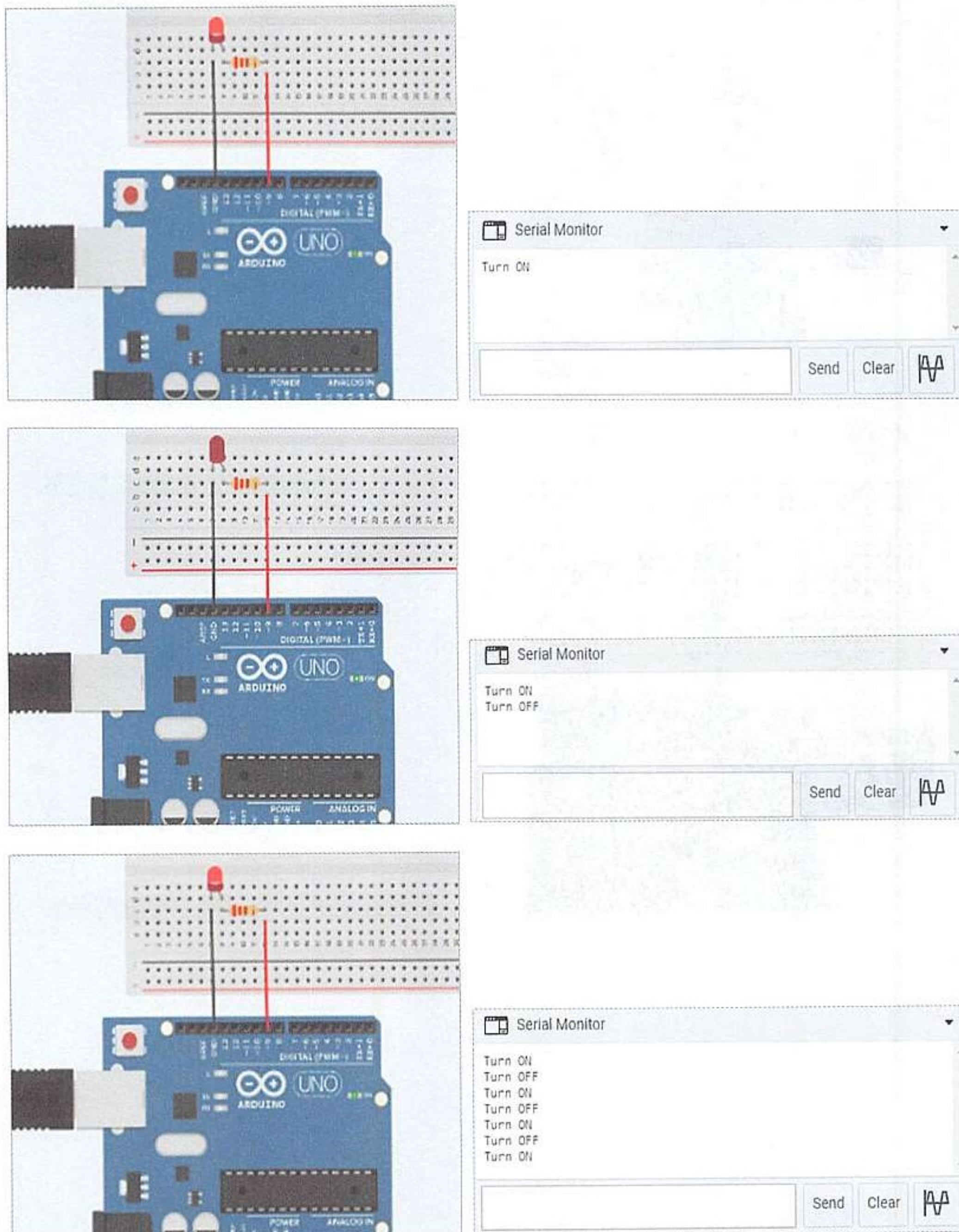
Arduino Board



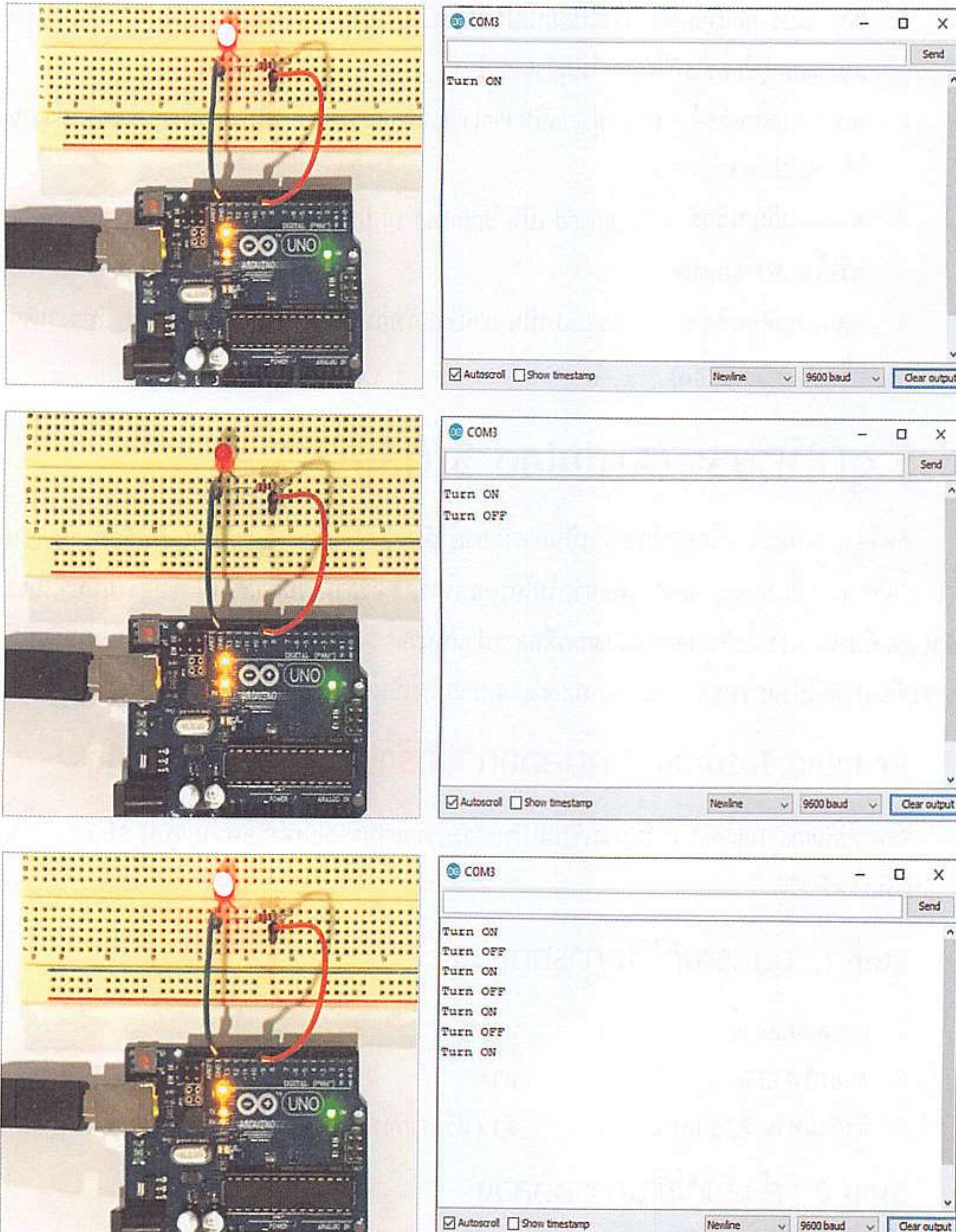
Step 6 : ผลการทดลอง Arduino Tutorial 4

จะเหมือนกับ Arduino Tutorial 2

Tinkercad Output



Arduino Board Output



Step 7 : คำถามท้าย Arduino Tutorial 4

1. ชื่อตัวแปร pinLed สามารถเปลี่ยนเป็นชื่ออื่นได้หรือไม่ ถ้าได้ควรจะเปลี่ยนเป็นชื่ออะไรที่เหมาะสม (ชื่ออะไรก็ได้ที่ทำให้ผู้อ่านเข้าใจ)
2. ทดลองเปลี่ยนชื่อตัวแปร pinLed เป็น PinLed เฉพาะบรรทัดที่ 1 โปรแกรมจะสามารถทำงานได้หรือไม่ เพราะเหตุใด
3. ทดลองเปลี่ยนชื่อตัวแปร pinLed เป็น 9pinLed ทุกบรรทัด โปรแกรมจะสามารถทำงานได้หรือไม่ เพราะเหตุใด
4. ทดลองเปลี่ยนชื่อตัวแปร pinLed เป็น pinLed9 ทุกบรรทัด โปรแกรมจะสามารถทำงานได้หรือไม่ เพราะเหตุใด

6.5 ชุดอักขระ (Arduino Strings)

สตริง (Strings) หรือ**ชุดอักขระ** เป็นการเรียงลำดับของตัวอักขระซึ่งรวมกันเป็นข้อความ เช่น คำว่า "Arduino" ซึ่ง Strings จะช่วยให้บอร์ดไมโครคอนโทรลเลอร์สามารถสื่อสารเป็นข้อความที่ทำให้เราเข้าใจได้ ที่ผ่านมาเราได้เห็นผลลัพธ์ของการสื่อสารผ่านหน้าจอ Serial Monitor มาแล้ว และในบทเรียนนี้เราจะพัฒนาตัวโปรแกรมของ Arduino Tutorial 4 มาสร้างเป็นตัวแปรเพื่อเก็บค่าข้อความ Strings

Arduino Tutorial 5 ทดลองการสร้างตัวแปรแบบ Strings

จาก Arduino Tutorial 4 ในการเขียนโปรแกรม เราสามารถแสดงข้อความแบบ Strings ผ่านตัวแปรที่เราสร้างได้

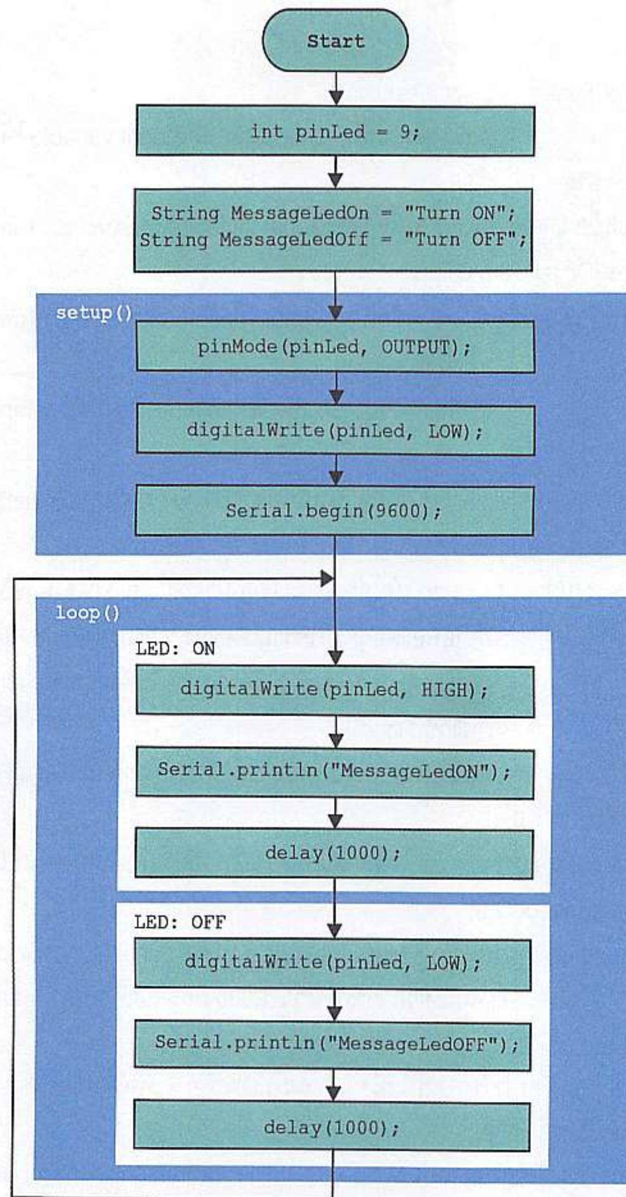
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1.	บอร์ด Arduino	1	บอร์ด
2.	หลอดไฟ LED	1	ดวง
3.	ตัวต้านทาน 220 โอห์ม	1	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 5

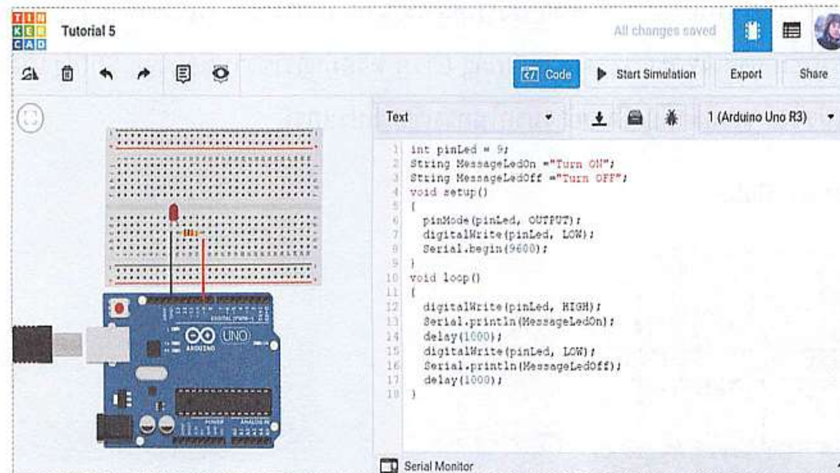


Step 4 : Source Code Arduino_Tutorial_5.ino

```
/*Tutorial 5
String Variables for Blink a LED with Arduino*/
int pinLed = 9;                 //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
String MessageLedOn = "Turn ON";
      //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ "Turn ON"
String MessageLedOff = "Turn OFF";
      //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ "Turn OFF"
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 pinMode(pinLed, OUTPUT);
      //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
 digitalWrite(pinLed, LOW);
      //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
 Serial.begin(9600);            //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //ฟังก์ชัน loop()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 digitalWrite(pinLed, HIGH);
      //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
 Serial.println(MessageLedOn);
      //แสดงผลข้อความที่เก็บในตัวแปร MessageLedOn บนจอ Serial Monitor และขึ้นบรรทัดใหม่
 delay(1000);                   //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 digitalWrite(pinLed, LOW);
      //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
 Serial.println(MessageLedOff);
      //แสดงผลข้อความที่เก็บในตัวแปร MessageLedOFF บนจอ Serial Monitor และขึ้นบรรทัดใหม่
 delay(1000);                   //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

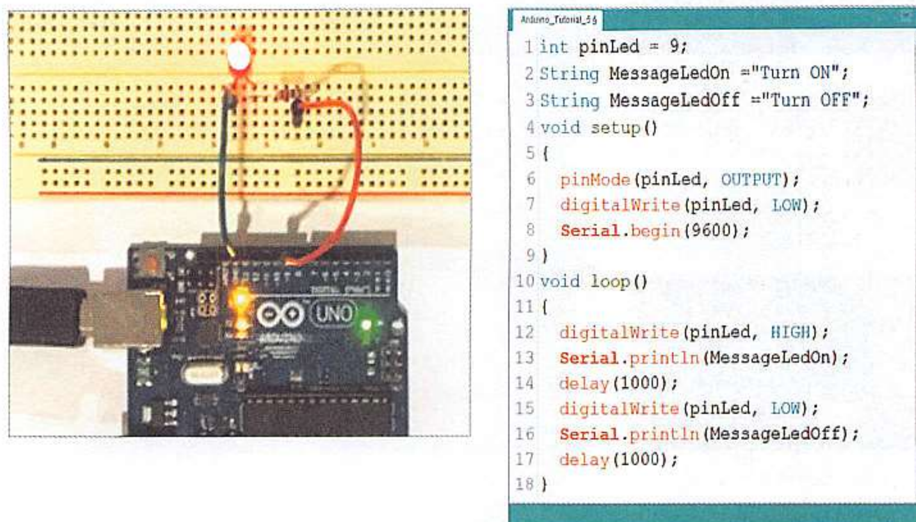
Step 5 : วิธีการทดลอง Arduino Tutorial 5

Tinkercad




```
int pinLed = 9;
String MessageLedOn ="Turn ON";
String MessageLedOff ="Turn OFF";
void setup()
{
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  Serial.begin(9600);
}
void loop()
{
  digitalWrite(pinLed, HIGH);
  Serial.println(MessageLedOn);
  delay(1000);
  digitalWrite(pinLed, LOW);
  Serial.println(MessageLedOff);
  delay(1000);
}
```

Arduino Board

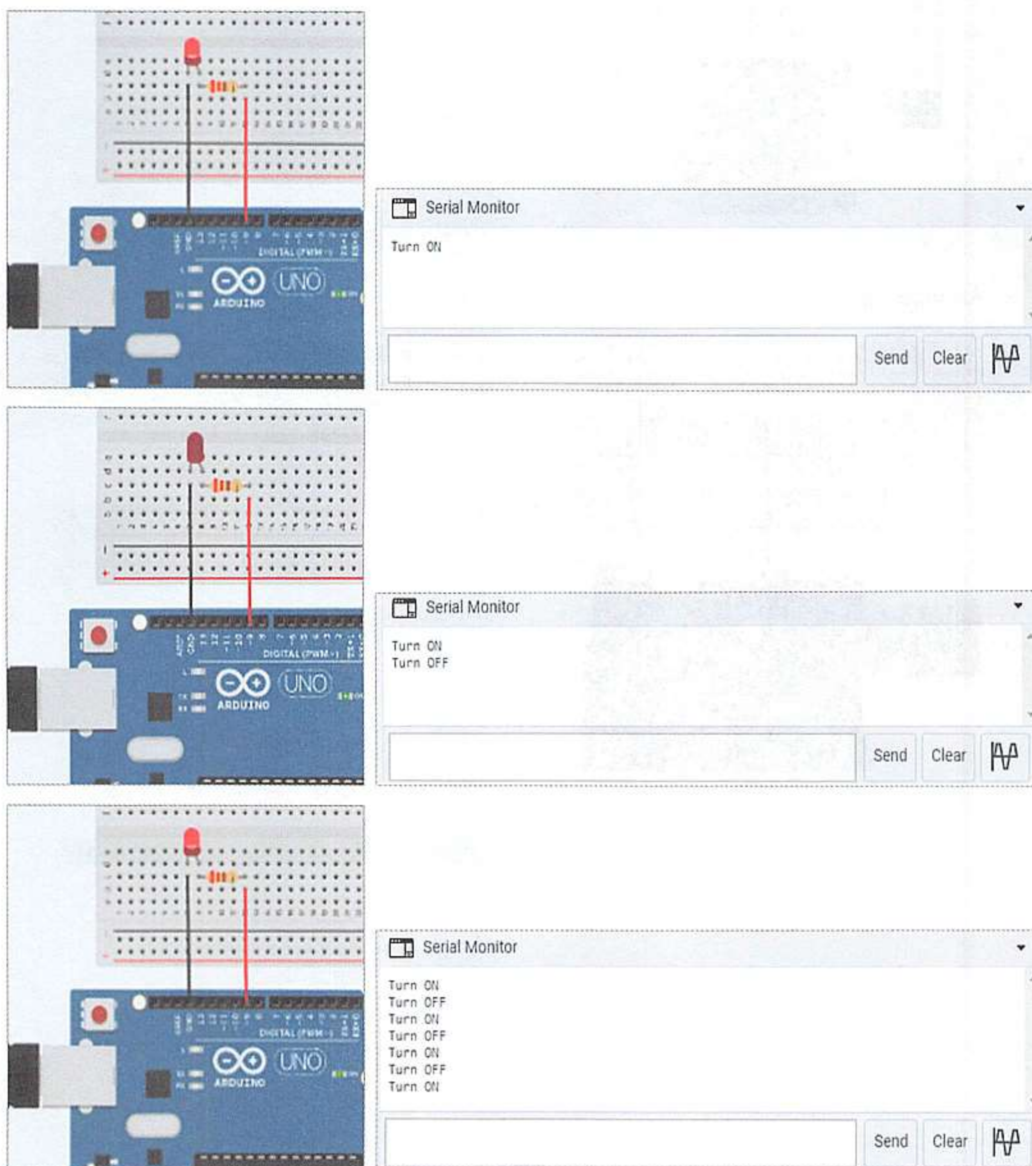



```
int pinLed = 9;
String MessageLedOn ="Turn ON";
String MessageLedOff ="Turn OFF";
void setup()
{
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  Serial.begin(9600);
}
void loop()
{
  digitalWrite(pinLed, HIGH);
  Serial.println(MessageLedOn);
  delay(1000);
  digitalWrite(pinLed, LOW);
  Serial.println(MessageLedOff);
  delay(1000);
}
```

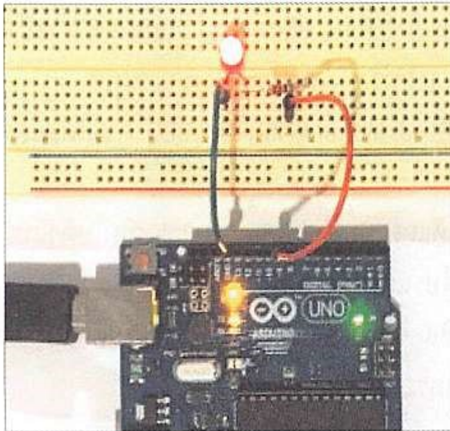
Step 6 : ผลการทดลอง Arduino Tutorial 5

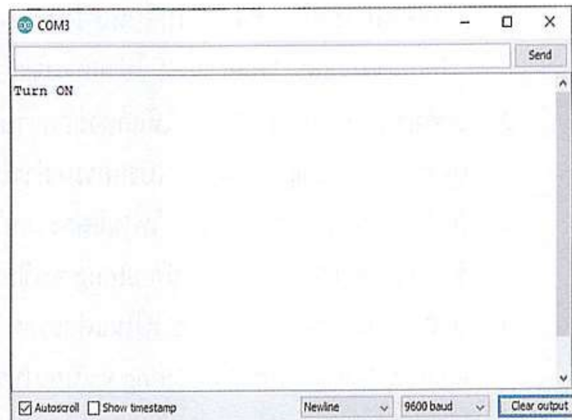
ผลการทดลองของ Arduino Tutorial 5 จะได้ผลเช่นเดียวกับ Arduino Tutorial 4 ด้วยการกำหนด String ให้กับตัวแปรแทนที่จะใช้โดยตรง จะทำให้ผู้พัฒนาโปรแกรมสามารถแก้ไข และเขียนโค้ดได้เป็นระบบและง่ายขึ้น โดยเฉพาะกับข้อความ String ยาวๆ หรือมีการใช้งานข้อความ String เดิมแบบซ้ำๆ การใช้ตัวแปรจึงมีความสำคัญเป็นอย่างมากในการเขียนโปรแกรม

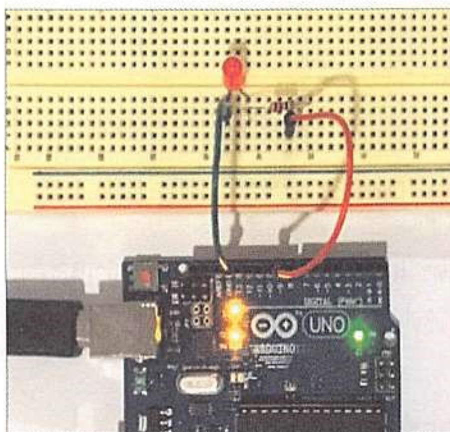
Tinkercad Output

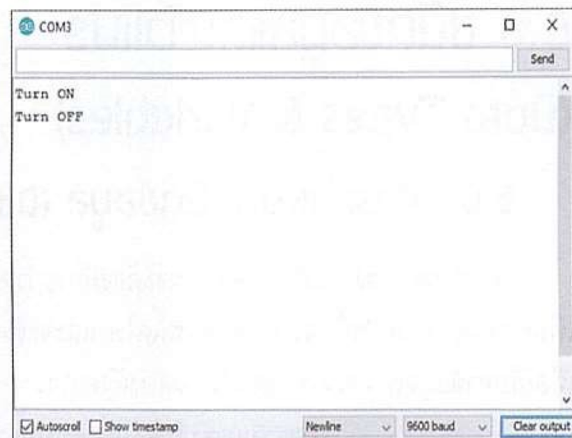


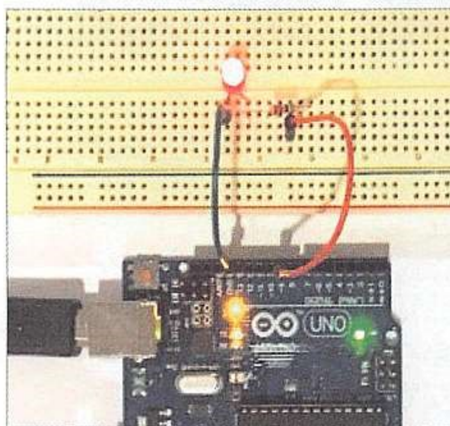
Arduino Board Output

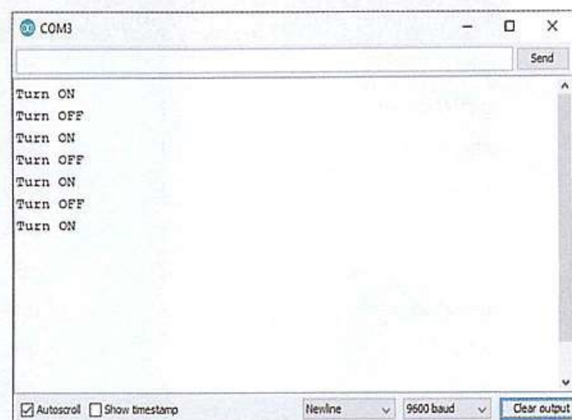












Step 7 : คำถามท้าย Arduino Tutorial 5

1. ทดลองแก้ไขโปรแกรมโดยเปลี่ยนเครื่องหมายจาก "Turn ON" และ "Turn OFF" เป็น 'Turn ON' และ 'Turn OFF' โปรแกรมจะสามารถทำงานได้หรือไม่ เพราะเหตุใด
2. ถ้าต้องการให้เมื่อ LED ติด ให้แสดงข้อความ Good Morning และเมื่อ LED ดับ ให้แสดงข้อความ Goodnight จะแก้ไขโปรแกรมอย่างไร
3. ถ้าต้องการให้ก่อนการกะพริบไฟในแต่ละรอบ ให้แสดงข้อความ Start Blink เก็บในตัวแปรชื่อ startLedMessage เป็นชนิด String จะเขียนโปรแกรมอย่างไร
4. ถ้าต้องการให้หลังการกะพริบไฟในแต่ละรอบ ให้แสดงข้อความ End Blink เก็บในตัวแปรชื่อ endLedMessage เป็นชนิด String จะเขียนโปรแกรมอย่างไร

6.6 ชนิดข้อมูลและตัวแปร (Data Types & Variables)

6.6.1 การกำหนดชนิดข้อมูล (Data Type)

ในการประกาศตัวแปรหรือฟังก์ชันจะต้องมีการกำหนดชนิดข้อมูล (Data Type) ให้กับตัวแปรด้วย เพื่อกำหนดว่าตัวแปรนั้นจะเก็บข้อมูลชนิดใด และจะต้องใช้พื้นที่ว่างเท่าใดในหน่วยความจำ (Size) เพื่อเก็บค่าข้อมูลเหล่านั้น รายละเอียดแสดงดังตาราง

ตารางแสดง Data Types บนแพลตฟอร์ม Arduino

ชนิดข้อมูล	ขนาด (ไบต์)	ช่วงของค่า
void	0	null
bool/boolean	1	True/False
char	1	-128 to +127
unsigned char	1	0 to 255
byte	1	0 to 255
int	2	-32,768 to 32,767
unsigned int	2	0 to 65,535
word	2	0 to 65,535
long	4	-2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295
float	4	-3.4028235E+38 to 3.4028235E+38
double	4	-3.4028235E+38 to 3.4028235E+38
string	-	character array

Void Type (void)

ชนิดข้อมูลวอยด์ เป็นชนิดข้อมูลมาตรฐานที่ใช้เพื่อประกาศฟังก์ชัน (Function Declaration) เท่านั้น โดยจะไม่มีการส่งคืนค่าใดๆ เมื่อถูกเรียกใช้ อีกทั้งยังเป็นส่วนสำคัญในการประกาศชื่อฟังก์ชันหลัก คือ ฟังก์ชัน setup และฟังก์ชัน loop

ตัวอย่างการใช้งาน

```
void setup()    {
  // พื้นที่สำหรับ setup code ซึ่งจะรันคำสั่งเพียงครั้งเดียว
}
void loop()    {
  // พื้นที่สำหรับ main code ซึ่งจะรันซ้ำไปเรื่อยๆ
}
```

Boolean Type (bool)

ชนิดข้อมูลบูลีน เป็นหนึ่งในชนิดข้อมูลมาตรฐานของ Arduino อีกเช่นกัน เพื่อเก็บค่าบูลีน (boolean value) ซึ่งก็คือ True (จริง สถานะ 1) หรือ False (เท็จ สถานะ 0) เช่น ใช้กำหนดชนิดตัวแปรเพื่อเก็บค่าบูลีนสำหรับควบคุมสถานะหลอดไฟ LED

ตัวอย่างการใช้งาน

```
bool Bool1 = true;
bool Bool2 = false;
```

Character Type (char)

ชนิดข้อมูลคาแรคเตอร์ เป็นชนิดข้อมูลที่ใช้เก็บค่าหมายเลขรหัสแอสกี้ (ASCII) ที่ประกอบด้วยอักษรละติน เลขอารบิก เครื่องหมายวรรคตอน และสัญลักษณ์ต่างๆ โดยแต่ละรหัสจะแทนด้วยตัวอักขระหนึ่งตัว และสามารถใช้เก็บอักขระ 1 ตัว โดยใส่ไว้ในเครื่องหมาย ' ' (Single Quote)

ตัวอย่างการใช้งาน

```
char Char1 = 65; // รหัส 65 (เลขฐานสิบ) ใช้แทนอักษร A พิมพ์ใหญ่
char Char2 = 'A' ; // กำหนดตัวแปรชื่อ Char2 เป็นข้อมูลชนิดคาแรคเตอร์เก็บอักษร A
```

ASCII Chart ใน Character

อักขระ 0-31 เป็นอักขระควบคุม Control Characters ที่ไม่สามารถพิมพ์ได้ และใช้เพื่อควบคุมอุปกรณ์ต่อพ่วง เช่น เครื่องพิมพ์ สำหรับอักขระ 32-127 เป็นอักขระที่พิมพ์ได้ (Printable Characters) แสดงถึงตัวอักษร ตัวเลข เครื่องหมาย วรรคตอน และสัญลักษณ์อื่นๆ

ตารางรหัส ASCII วิธีมาตรฐานในการเข้ารหัสอักขระเป็นตัวเลข

Dec	Hex	Char	Name/Function
0	0	NUL	Null
1	1	SOH	Start Of Heading
2	2	STX	Start Of Text
3	3	ETX	End Of Text
4	4	EOT	End Of Transmit
5	5	ENQ	Enquiry
6	6	ACK	Acknowledge
7	7	BEL	Bell
8	8	BS	Backspace
9	9	HT	Horizontal Tab
10	0A	LF	Line Feed
11	0B	VT	Vertical Tab
12	0C	FF	Form Feed
13	0D	CR	Carriage Return
14	0E	SO	Shift Out
15	0F	SI	Shift In
16	10	DLE	Data Line Escape
17	11	DC1	Device Control 1
18	12	DC2	Device Control 2
19	13	DC3	Device Control 3
20	14	DC4	Device Control 4
21	15	NAK	Non Acknowledge
22	16	SYN	Synchronous Idle
23	17	ETB	End Transmit Block
24	18	CAN	Cancel
25	19	EM	End Of Medium
26	1A	SUB	Substitute
27	1B	ESC	Escape
28	1C	FS	File Separator
29	1D	GS	Group Separator
30	1E	RS	Record Separator
31	1F	US	Unit Separator

Reference : https://learn.parallax.com/support/reference/ascii-table-0-127

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	space	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	delete

หรือ https://www.arduino.cc/en/Reference/ASCIIchart

Unsigned Character/Byte Type (unsigned char/byte)

ชนิดข้อมูลอันไซน์คาแรคเตอร์/ไบต์ ใน Arduino ข้อมูลทั้ง 2 ชนิดนี้จะเหมือนกัน คือ ใช้เก็บข้อมูลตัวเลขที่ไม่คิดเครื่องหมาย (ไม่มีเครื่องหมายลบ) ขนาด 1 byte หรือ 8 bits เก็บค่าได้ตั้งแต่ 0-255

ตัวอย่างการใช้งาน

```
unsigned char UsChar1 = 0;
unsigned char UsChar2 = 255;
byte Byte1 = 0;
byte Byte2 = 255;
```

Integer Type (int)

ชนิดข้อมูลอินทิเจอร์ เป็นข้อมูลชนิดจำนวนเต็มที่นิยมใช้มากที่สุดใน Arduino โดยบอร์ดตระกูล ATmega เช่น UNO, MEGA, NANO และอื่นๆ ข้อมูลแบบ int จะใช้พื้นที่หน่วยความจำขนาด 2 bytes ในช่วง -32,768 ถึง +32,767 แต่ในบอร์ดระดับสูงบางรุ่น เช่น DUE และ MKR1000 ข้อมูลแบบ int จะใช้พื้นที่หน่วยความจำ 4 bytes ในช่วง -2,147,483,648 ถึง +2,147,483,647

ตัวอย่างการใช้งาน

```
int Int1 = 65;
int Int2 = -69;
```

Unsigned Integer Type (unsigned int/word)

อันไซน์อินทิเจอร์ และเวิร์ด จะเหมือนกันกับชนิดข้อมูล int คือ ใช้เก็บค่าของตัวแปรที่เป็นจำนวนเต็มบวก ไม่เก็บค่าจำนวนเต็มลบ ดังนั้น ช่วงของค่าจึงกลายเป็น 0 ถึง 65,535 เช่นเดียวกับ Integer ของบอร์ดระดับสูงบางรุ่น เช่น Arduino DUE และ MKR1000 ที่ใช้พื้นที่หน่วยความจำขนาด 4 bytes และช่วงจะเริ่มที่ 0 ถึง 4,294,967,295

ตัวอย่างการใช้งาน

```
unsigned int UsInt1 = -23;
unsigned int UsInt2 = 2800;
word Word1 = 0;
word Word2 = 899;
```

Long Type (long)

ลอง เป็นชนิดข้อมูลจำนวนเต็ม ใช้พื้นที่หน่วยความจำขนาด 4 bytes ทำให้เก็บข้อมูลได้มากกว่าข้อมูลชนิด int ที่ใช้เพียง 2 bytes โดยจะเก็บค่าอยู่ในช่วง -2,147,483,648 ถึง 2,147,483,647

ตัวอย่างการใช้งาน

```
long Long1 = -989923;
long Long2 = 2147483647;
```

Unsigned Long Type (unsigned long)

อันไซน์ลอง เป็นชนิดข้อมูลจำนวนเต็มแต่ไม่เก็บจำนวนเต็มลบ ใช้พื้นที่หน่วยความจำขนาด 4 bytes ในการเก็บข้อมูล โดยจะเก็บค่าอยู่ในช่วง 0 ถึง 4,294,967,295

ตัวอย่างการใช้งาน

```
unsigned long UsLong1 = 2345689;
unsigned long UsLong2 = 4294967295;
```

Float และ Double Type (float/double)

ชนิดข้อมูลโฟลต/ดับเบิล เป็นชนิดข้อมูลสำหรับเก็บค่าตัวเลขทศนิยม ในบอร์ดตระกูล Atmega (Arduino Uno และอื่นๆ) ชนิดข้อมูลทั้ง 2 จะเหมือนกัน โดยเลขทศนิยมจะใช้เพื่อประมาณค่าแบบอะนาล็อกและค่าแบบต่อเนื่อง เนื่องจากมีความละเอียดมากกว่าจำนวนเต็ม ใช้พื้นที่หน่วยความจำขนาด 4 bytes เก็บข้อมูลในช่วง -3.4028235E+38 ถึง 3.4028235E+38 เช่นเดียวกัน อย่างไรก็ตาม ในบอร์ดบางรุ่น เช่น Arduino DUE ชนิดข้อมูล double จะใช้หน่วยความจำ 8 bytes

ตัวอย่างการใช้งาน

```
float Float1 = -8.119;
float Float2 = 23.89;
double DoubleNum1 = -8.119;
double DoubleNum2 = 23.89;
```

String Type

ชนิดข้อมูลสตริง เป็นชนิดตัวแปรแบบข้อความ หรือเรียกว่า "อาร์เรย์ของตัวอักขระ (Character Array)" ที่แสดงในบทเรียน Arduino Tutorial 5 ผ่านมา

ตัวอย่างการใช้งาน

```
String MessageLedOn = "Turn ON";
String MessageLedOff = "Turn OFF";
```

6.6.2 อักขระหลีก (Escape Character)

อักขระหลีก (Escape Character) คือ ตัวอักขระบนแป้นพิมพ์ที่เมื่อกดแล้ว ทำให้ตัวอักขระอื่นๆ ที่ตามมา ต้องแปลความต่างไปจากตัวอักขระที่ใช้มาก่อนหน้านี้ ตัวอย่างเช่น จะใช้อักขระหลีกเพื่อกำหนดความหมายว่า ตัวอักขระที่จะใช้ต่อไปนั้น จะต้องใช้รหัสแปลความแตกต่างไปจากรหัสแปลความที่เคยใช้ก่อนหน้านี้ หรืออาจจะถือว่าอักขระเหล่านั้นไม่ใช่ข้อมูลอีกต่อไป แต่เป็นอักขระเพื่อการควบคุมเท่านั้น เป็นต้น

เนื่องจากเครื่องหมายหรืออักขระบางอย่าง เราไม่สามารถเขียนโค้ดเพื่อแสดงผลอย่างตรงไปตรงมาได้ เช่น เครื่องหมายคำพูด “” (Double Quote) ที่ใช้บอกว่าข้อความ หรือ String เริ่มต้นและสิ้นสุดที่อักษรตัวไหน ดังนั้น เมื่อเจอเครื่องหมาย ” ในสายอักขระ คอมพิวเตอร์จะแปลความว่าสิ้นสุดสายอักขระนั้นแล้ว เช่นเดียวกับเครื่องหมาย ‘’ (Single Quote) และ \ (Backslash) หรืออักขระบางตัว เราไม่สามารถพิมพ์จากแป้นคีย์บอร์ดได้เพราะว่ามันไม่มี เช่น อักขระขึ้นบรรทัดใหม่ อักขระแท็บ เป็นต้น เราจึงต้องใช้ตัวอักขระพิเศษที่เรียกว่า อักขระหลีก คือ การนำเครื่องหมาย \ มาวางไว้หน้าอักขระที่กำหนด จะมีผลทำให้อักขระที่ตามหลังเครื่องหมาย \ มีความหมายเปลี่ยนไปจากเดิม ดังรูป

ตาราง Escape Character พื้นฐาน

Escape Sequence	Description
\n	ขึ้นบรรทัดใหม่
\t	เว้นวรรค 1 Tab
\\	แสดงเครื่องหมาย \
\’	แสดงเครื่องหมาย ’

Arduino Tutorial 6 ทดลองการใช้งาน Data Types และ Escape Characters

ในการทดลองนี้ เราจะได้เรียนรู้การใช้งานชนิดข้อมูลและการแสดงผลของชนิดข้อมูลแบบต่างๆ

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

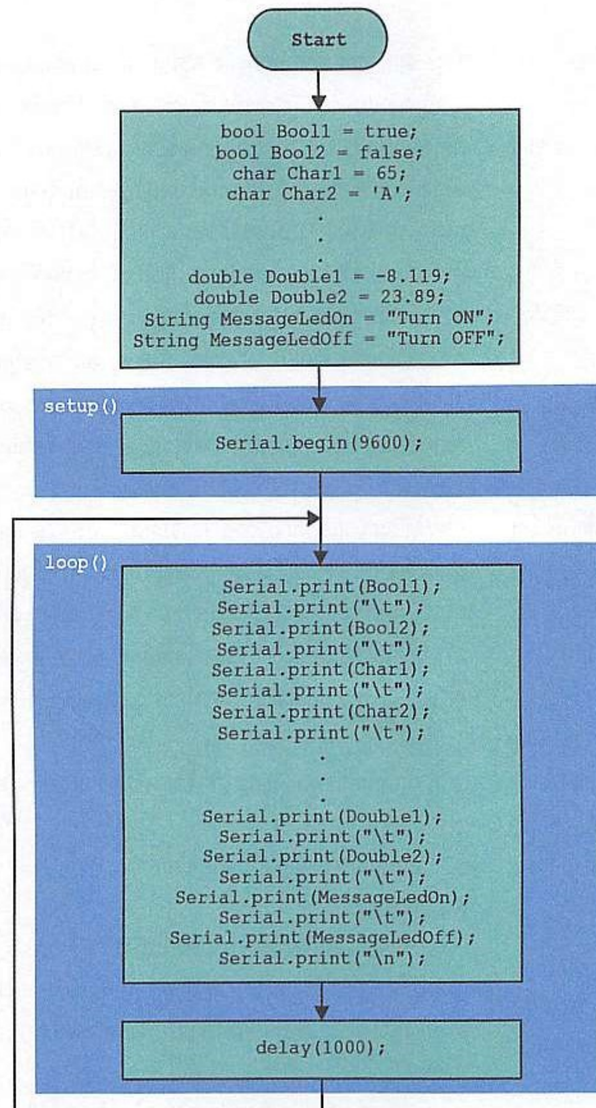
บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

setup(): Serial.begin()

loop(): Serial.print(), Serial.println(), delay()

Step 3 : Flowchart Arduino Tutorial 6



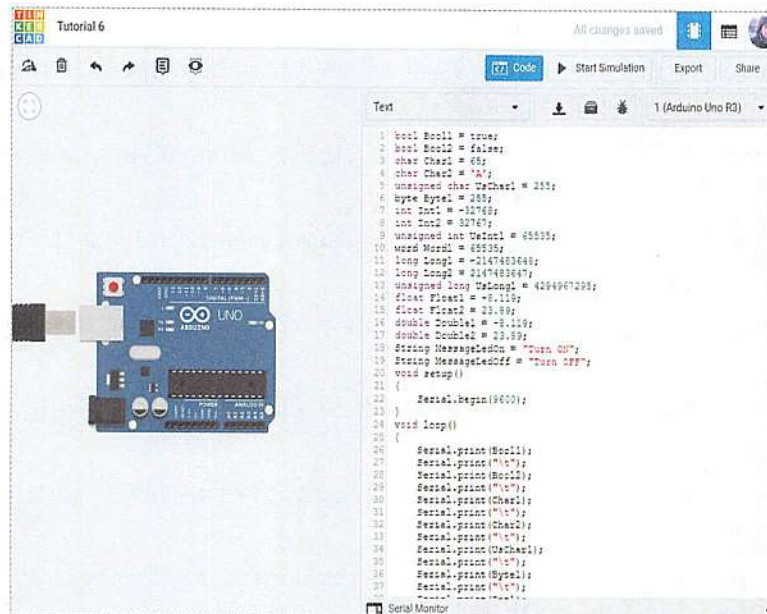
Step 4 : Source Code Arduino_Tutorial_6.ino

```
/* Arduino Tutorial 6
  Data Types*/
bool Bool1 = true;            //สร้างตัวแปรชื่อ Bool1 เป็นชนิดข้อมูล bool เก็บข้อมูล true มีค่าเท่ากับ 1
bool Bool2 = false;           //สร้างตัวแปรชื่อ Bool2 เป็นชนิดข้อมูล bool เก็บข้อมูล false มีค่าเท่ากับ 0
char Char1 = 65;
      //สร้างตัวแปรชื่อ Char1 เป็นชนิดข้อมูล char เก็บข้อมูลรหัส ASCII มีค่าตัวเลขเท่ากับ 65 คือ ตัวอักขระ A
char Char2 = 'A';              //สร้างตัวแปรชื่อ Char2 เป็นชนิดข้อมูล char เก็บข้อมูลตัวอักขระ A
unsigned char UsChar1 = 255;  //สร้างตัวแปรชื่อ UsChar1 เป็นชนิดข้อมูล unsigned char เก็บข้อมูลตัวเลข 255
byte Byte1 = 255;             //สร้างตัวแปรชื่อ Byte1 เป็นชนิดข้อมูล byte เก็บข้อมูลตัวเลข 255
int Int1 = -32768;            //สร้างตัวแปรชื่อ Int1 เป็นชนิดข้อมูล int เก็บข้อมูลตัวเลข -32768
int Int2 = 32767;             //สร้างตัวแปรชื่อ Int2 เป็นชนิดข้อมูล int เก็บข้อมูลตัวเลข 32767
unsigned int UsInt1 = 65535;  //สร้างตัวแปรชื่อ UsInt1 เป็นชนิดข้อมูล unsigned int เก็บข้อมูลตัวเลข 65535
word Word1 = 65535;           //สร้างตัวแปรชื่อ Word1 เป็นชนิดข้อมูล word เก็บข้อมูลตัวเลข 65535
long Long1 = -2147483648;     //สร้างตัวแปรชื่อ Long1 เป็นชนิดข้อมูล long เก็บข้อมูลตัวเลข -2147483648
long Long2 = 2147483647;      //สร้างตัวแปรชื่อ Long2 เป็นชนิดข้อมูล long เก็บข้อมูลตัวเลข 2147483647
unsigned long UsLong1 = 4294967295;
      //สร้างตัวแปรชื่อ UsLong1 เป็นชนิดข้อมูล unsigned long เก็บข้อมูลตัวเลข 4294967295
float Float1 = -8.119;        //สร้างตัวแปรชื่อ Float1 เป็นชนิดข้อมูล float เก็บข้อมูลตัวเลข -8.119
float Float2 = 23.89;         //สร้างตัวแปรชื่อ Float2 เป็นชนิดข้อมูล float เก็บข้อมูลตัวเลข 23.89
double Double1 = -8.119;      //สร้างตัวแปรชื่อ Double1 เป็นชนิดข้อมูล double เก็บข้อมูลตัวเลข -8.119
double Double2 = 23.89;       //สร้างตัวแปรชื่อ Double2 เป็นชนิดข้อมูล double เก็บข้อมูลตัวเลข 23.89
String MessageLedOn = "Turn ON";
      //สร้างตัวแปรชื่อ MessageLedOn เป็นชนิดข้อมูล String เก็บข้อมูลข้อความ Turn ON
String MessageLedOff = "Turn OFF";
      //สร้างตัวแปรชื่อ MessageLedOff เป็นข้อมูล String เก็บข้อมูลข้อความ Turn OFF
void setup()                  //ฟังก์ชัน setup()
{                             //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  Serial.begin(9600);         //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                             //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                   //ฟังก์ชัน loop()
{                             //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  Serial.print(Bool1);        //แสดงค่าข้อมูลของตัวแปร Bool1 ออกทางหน้าจอ Serial Monitor
  Serial.print("\t");         //เว้นวรรค 1 Tab
  Serial.print(Bool2);        //แสดงค่าข้อมูลของตัวแปร Bool2 ออกทางหน้าจอ Serial Monitor
  Serial.print("\t");         //เว้นวรรค 1 Tab
```

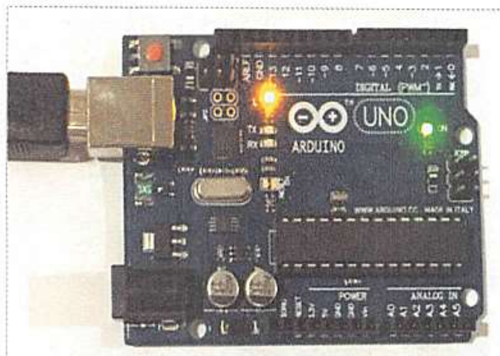
```
    Serial.print(Char1);          //แสดงค่าข้อมูลของตัวแปร Char1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Char2);          //แสดงค่าข้อมูลของตัวแปร Char2 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(UsChar1);        //แสดงค่าข้อมูลของตัวแปร UsChar1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Byte1);          //แสดงค่าข้อมูลของตัวแปร Byte1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Int1);           //แสดงค่าข้อมูลของตัวแปร Int1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Int2);           //แสดงค่าข้อมูลของตัวแปร Int2 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(UsInt1);         //แสดงค่าข้อมูลของตัวแปร UsInt1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Word1);          //แสดงค่าข้อมูลของตัวแปร Word1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Long1);          //แสดงค่าข้อมูลของตัวแปร Long1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Long2);          //แสดงค่าข้อมูลของตัวแปร Long2 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(UsLong1);        //แสดงค่าข้อมูลของตัวแปร UsLong1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Float1);         //แสดงค่าข้อมูลของตัวแปร Float1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Float2);         //แสดงค่าข้อมูลของตัวแปร Float2 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Double1);        //แสดงค่าข้อมูลของตัวแปร Double1 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(Double2);        //แสดงค่าข้อมูลของตัวแปร Double2 ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(MessageLedOn);   //แสดงค่าข้อมูลของตัวแปร MessageLedOn ออกทางหน้าจอ Serial Monitor
    Serial.print("\t");           //เว้นวรรค 1 Tab
    Serial.print(MessageLedOff);  //แสดงค่าข้อมูลของตัวแปร MessageLedOff ออกทางหน้าจอ Serial Monitor
    Serial.print("\n");           //ขึ้นบรรทัดใหม่
    delay(1000);                  //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                                 //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

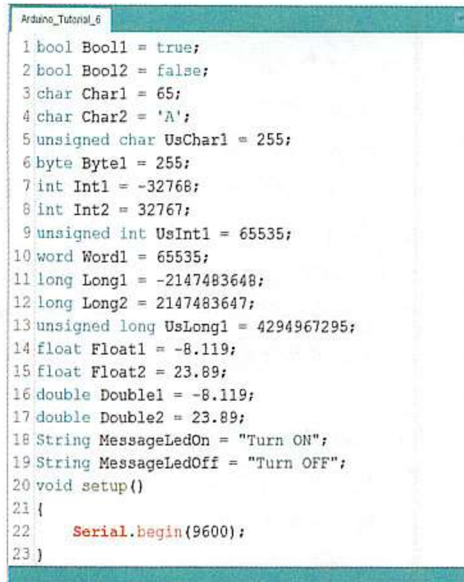
Step 5 : วิธีการทดลอง Arduino Tutorial 6

Tinkercad



Arduino Board




Arduino_Tutorial_6

```
bool Bool1 = true;
bool Bool2 = false;
char Char1 = 65;
char Char2 = 'A';
unsigned char UsChar1 = 255;
byte Byte1 = 255;
int Int1 = -32768;
int Int2 = 32767;
unsigned int UsInt1 = 65535;
word Word1 = 65535;
long Long1 = -2147483648;
long Long2 = 2147483647;
unsigned long UsLong1 = 4294967295;
float Float1 = -8.119;
float Float2 = 23.89;
double Double1 = -8.119;
double Double2 = 23.89;
String MessageLedOn = "Turn ON";
String MessageLedOff = "Turn OFF";
void setup()
{
    Serial.begin(9600);
}
```

Step 6 : ผลการทดลอง Arduino Tutorial 6

จะแสดงค่าข้อมูลของชนิดตัวแปรแต่ละตัวเรียงตามลำดับ ดังนี้

Bool1, Bool2, Char1, Char2, UsChar1, Byte1, Int1, Int2, UsInt1, Word1, Long1, Long2, UsLong1, Float1, Float2, Double1, Double2, MessageLedOn, MessageLedOff และจะแสดงผลค่าข้อมูลของแต่ละตัวแปรวนซ้ำไปเรื่อยๆ

Tinkercad Output

```
Serial Monitor
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
```

Arduino Board Output

```
COM3                                                                                          Send
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
1   0   A   A   255   255   -32768  32767   65535   65535   -2147483648   2147483647   4294967295   -8.12   23.89   -8.12   23.89   Turn ON Turn OFF
Autoscroll  Show timestamp                                         Newline   9600 baud   Clear output
```

Step 7 : คำถามท้าย Arduino Tutorial 6

1. ทดลองเปลี่ยนค่าตัวแปร Char1 จากเดิม 65 เปลี่ยนเป็น 89 ผลลัพธ์จะได้ตัวอักขระอะไร เพราะเหตุใด
2. ทดลองเปลี่ยนค่าตัวแปร Byte1 จากเดิม 255 เปลี่ยนเป็น 256 และ 259 ผลลัพธ์จะเป็นอย่างไร เพราะเหตุใด
3. ทดลองเปลี่ยนค่าตัวแปร Int2 จากเดิม 32767 เปลี่ยนเป็น 32767.89 ผลลัพธ์จะเป็นอย่างไร ถ้าต้องการให้แสดงค่าทศนิยมจะแก้ไขโปรแกรมอย่างไร
4. ทดลองเพิ่มตัวแปรชื่อ Arduino โดยเขียนโปรแกรมให้แสดงข้อความว่า “I love Arduino Programming” จะเขียนโปรแกรมอย่างไร

6.7 โอเปอเรเตอร์ (Arduino Operators)

โอเปอเรเตอร์ (Operators) คือ ตัวดำเนินการที่แทนด้วยสัญลักษณ์เพื่อบอกให้คอมไพเลอร์ใช้ฟังก์ชันทางคณิตศาสตร์ หรือตรรกะมาใช้ในการเขียนโปรแกรม ภาษาโค้ดที่ใช้โดย Arduino จะมีทั้งหมด 5 แบบ ได้แก่ ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators), ตัวดำเนินการเปรียบเทียบ (Comparison Operators), ตัวดำเนินการทางตรรกะ (Boolean Operators), ตัวดำเนินการแบบบิตไวส์ (Bitwise Operators) และตัวดำเนินการกำหนดค่าแบบผสม (Compound Operators)

6.7.1 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

กำหนดให้ตัวแปร a เก็บค่า 20 และตัวแปร b เก็บค่า 10

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	ตัวอย่าง
Assignment Operator	=	การเก็บค่าข้อมูลทางด้านขวาของเครื่องหมายเท่ากับ คือ ตัวแปร b มาไว้ในตัวแปรทางด้านซ้ายของเครื่องหมายเท่ากับ คือ ตัวแปร a	a = b
Addition	+	การบวกค่าตัวถูกดำเนินการ โดยนำค่าข้อมูลของตัวแปร a มาบวกกับค่าข้อมูลของตัวแปร b	a + b จะได้ 30
Subtraction	-	การลบค่าตัวถูกดำเนินการ โดยนำค่าข้อมูลของตัวแปร a มาลบกับค่าข้อมูลของตัวแปร b	a - b จะได้ 10
Multiplication	*	การคูณค่าตัวถูกดำเนินการ โดยนำค่าข้อมูลของตัวแปร a มาคูณกับค่าข้อมูลของตัวแปร b	a * b จะได้ 200
Division	/	การหารค่าตัวถูกดำเนินการ โดยนำค่าข้อมูลของตัวแปร a มาหารกับค่าข้อมูลของตัวแปร b	a / b จะได้ 2
Modulo	%	การหารค่าตัวถูกดำเนินการแล้วเอาเฉพาะเศษ โดยนำค่าข้อมูลของตัวแปร a มาหารกับค่าข้อมูลของตัวแปร b แล้วจะใช้ตัวเศษ	a % b จะได้ 0

6.7.1.1 ลำดับความสำคัญของตัวดำเนินการ

หมวด	สัญลักษณ์โอเปอเรเตอร์	ลำดับ
Postfix	() [] -> . ++ - -	Left to Right
Unary	+ - ! ~ ++ - - (type) * & sizeof	Right to Left
Multiplicative	* / %	Left to Right
Additive	+ -	Left to Right
Shift	<< >>	Left to Right
Relational	< ⩽ > ⩾	Left to Right
Equality	== !=	Left to Right
Bitwise AND	&	Left to Right
Bitwise XOR	^	Left to Right
Bitwise OR	|	Left to Right
Logical AND	&&	Left to Right
Logical OR	||	Left to Right
Conditional	?:	Right to Left
Assignment	= += -= *= /= %= >⩾ <⩽ &= ^= |=	Right to Left
Comma	,	Left to Right

Arduino Tutorial 7 ทดลองใช้งานตัวดำเนินการทางคณิตศาสตร์

แล็บการทดลองนี้ เราจะได้เรียนรู้การใช้งานตัวดำเนินการทางคณิตศาสตร์

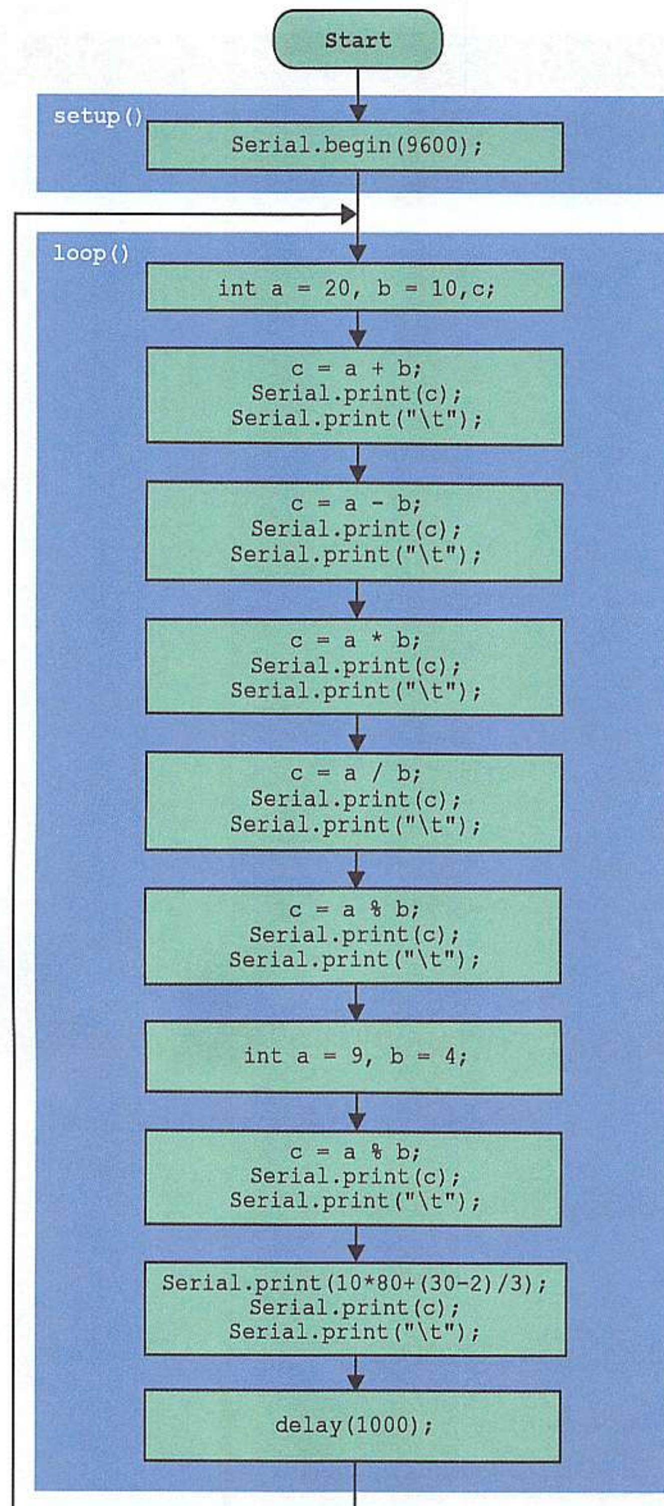
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): Serial.begin()
loop():  Serial.print(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 7

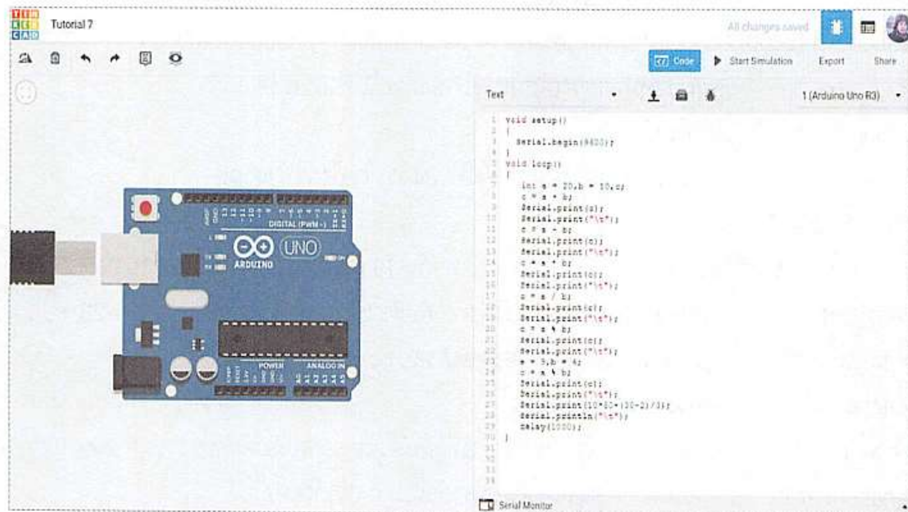


Step 4 : Source Code Arduino_Tutorial_7.ino

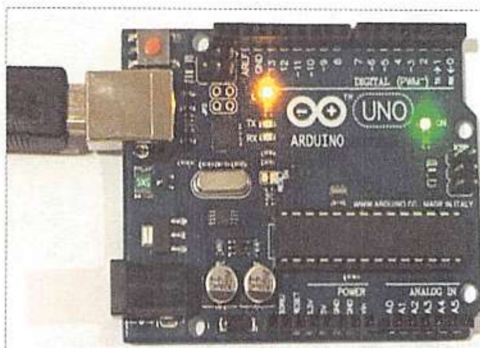
```
void setup()           //ฟังก์ชัน setup()
{                      //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);   //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                      //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()            //ฟังก์ชัน loop()
{                      //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  int a = 20,b = 10,c;
       //กำหนดให้ตัวแปร a เท่ากับ 20 ตัวแปร b เท่ากับ 10 และตัวแปร c เก็บเลขจำนวนเต็ม
  c = a + b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a บวกกับตัวแปร b (c = 20 + 10 = 30)
  Serial.print(c);     //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 30
  Serial.print("\t");  //เว้นวรรค 1 Tab
  c = a - b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a ลบกับตัวแปร b (c = 20 - 10 = 10)
  Serial.print(c);     //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 10
   Serial.print("\t"); //เว้นวรรค 1 Tab
  c = a * b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a คูณกับตัวแปร b (c = 20 * 10 = 200)
  Serial.print(c);     //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 200
  Serial.print("\t");  //เว้นวรรค 1 Tab
  c = a / b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a หารกับตัวแปร b (c = 20 / 10 = 10)
  Serial.print(c);     //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 10
  Serial.print("\t");  //เว้นวรรค 1 Tab
  c = a % b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a มอดกับตัวแปร b (c = 20 % 10 = 0)
  Serial.print(c);
       //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 0 (มอด คือ การหารเอาเศษ หารลงตัวเศษ 0)
  Serial.print("\t");  //เว้นวรรค 1 Tab
  a = 9,b = 4;         //กำหนดให้ตัวแปร a เท่ากับ 9 และตัวแปร b เท่ากับ 4
  c = a % b;           //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a มอดกับตัวแปร b (c = 9 % 4 = 1)
  Serial.print(c);     //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 1 (มอด : การหารเอาเศษ หารเหลือเศษ 1)
  Serial.print("\t");  //เว้นวรรค 1 Tab
  Serial.print(10*80+(30-2)/3);
       //แสดงค่าข้อมูลผลลัพธ์การคำนวณตามลำดับความสำคัญของตัวดำเนินการ
  Serial.println("\t"); //เว้นวรรค 1 Tab
  delay(1000);         //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                      //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

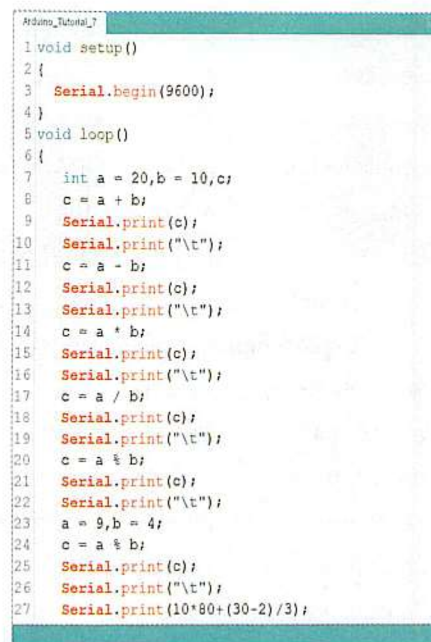
Step 5 : วิธีการทดลอง Arduino Tutorial 7

Tinkercad



Arduino Board




Arduino_Tutorial_7

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
   int a = 20,b = 10,c;
   c = a + b;
   Serial.print(c);
   Serial.print("\t");
   c = a - b;
   Serial.print(c);
   Serial.print("\t");
   c = a * b;
   Serial.print(c);
   Serial.print("\t");
   c = a / b;
   Serial.print(c);
   Serial.print("\t");
   c = a % b;
   Serial.print(c);
   Serial.print("\t");
   a = 9,b = 4;
   c = a % b;
   Serial.print(c);
   Serial.print("\t");
   Serial.print(10*80+(30-2)/3);
```

Step 6 : ผลการทดลอง Arduino Tutorial 7

ผลลัพธ์ที่ได้จะแสดงการคำนวณเรียงตามตัวดำเนินการ ดังนี้

ตั้งค่า a = 20 และ b = 10

c = a + b = 20 + 10 = 30

c = a – b = 20 – 10 = 10

c = a * b = 20 * 10 = 200

c = a / b = 20 / 10 = 2

c = a % b = 20/10 = 2 เศษ 0 (หารเอาเศษ)

เปลี่ยนค่า a = 9 และ b = 4 จะได้

c = a % b = 9 % 4 = 9/4 = 2 เศษ 1 (หารเอาเศษ)

คำนวณค่า 10 * 80 + (30-2)/3 วิธีคิดจะคำนวณตามลำดับความสำคัญของตัวดำเนินการในข้อนี้ คือ () * / + – เรียงจากซ้ายไปขวา จะได้

10 * 80 + (30-2)/3

= 10 * 80 + (28)/3

= 800 + 28/3

= 800 + 9.33

= 809 เนื่องจาก Default จะแสดงเป็นเลขจำนวนเต็ม

Tinkercad Output

Serial Monitor

```
30    10    200    2    0    1    809
30    10    200    2    0    1    809
30    10    200    2    0    1    809
```

Arduino Board Output

COM3

```
30    10    200    2    0    1    809
30    10    200    2    0    1    809
30    10    200    2    0    1    809
```

Step 7 : คำถามท้าย Arduino Tutorial 7

1. ทดลองแก้ไขโปรแกรมโดยกำหนดให้ a = 30 และ b = 89 ผลลัพธ์ที่ได้จะเป็นอย่างไร
2. ทดลองเขียนโปรแกรมเพื่อคำนวณค่า 23.89*93/60+(290-10)%9 ให้แสดงผลลัพธ์ออกทาง Serial Monitor
3. ทดลองเขียนโปรแกรมคำนวณหาพื้นที่สี่เหลี่ยมผืนผ้า กำหนดให้ตัวแปร width = 30 ตัวแปร height = 90 และตัวแปร area เก็บค่าผลลัพธ์ โดยให้แสดงผลลัพธ์ออกทาง Serial Monitor
4. ทดลองเขียนโปรแกรมแปลงค่าอุณหภูมิจากองศาฟาเรนไฮต์ (F) เป็นองศาเซลเซียส (C) ด้วยสมการ C = 5*(F-32)/9 กำหนดให้ตัวแปร F คือ องศาฟาเรนไฮต์มีค่าเท่ากับ 128 ให้แสดงผลลัพธ์ค่า C ออกทาง Serial Monitor

6.7.2 ตัวดำเนินการเปรียบเทียบ (Comparison Operators)

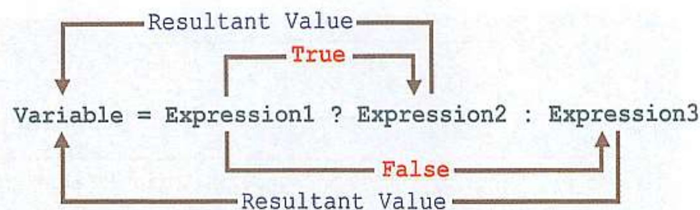
กำหนดให้ตัวแปร a เก็บค่า 10 และตัวแปร b เก็บค่า 20

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	ตัวอย่าง
equal to	==	ตรวจสอบว่าค่าของตัวแปร a มีค่าเท่ากับค่าของตัวแปร b หรือไม่ ถ้าเท่ากันเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้าไม่เท่ากันเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a == b) มีค่าเป็นเท็จ คือ 0 (False)
not equal to	!=	ตรวจสอบว่าค่าของตัวแปร a มีค่าเท่ากับค่าของตัวแปร b หรือไม่ ถ้าไม่เท่ากันเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่าเท่ากันเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a != b) มีค่าเป็นจริง คือ 1 (True)
less than	<	ตรวจสอบว่าค่าของตัวแปร a มีค่าน้อยกว่าค่าของตัวแปร b หรือไม่ ถ้าน้อยกว่าเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่ามากกว่าหรือเท่ากับเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a < b) มีค่าเป็นจริง คือ 1 (True)
greater than	>	ตรวจสอบว่าค่าของตัวแปร a มีค่ามากกว่าค่าของตัวแปร b หรือไม่ ถ้ามากกว่าเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่าน้อยกว่าหรือเท่ากับเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a > b) มีค่าเป็นเท็จ คือ 0 (False)
less than or equal to	⩽	ตรวจสอบว่าค่าของตัวแปร a มีค่าน้อยกว่าหรือเท่ากับค่าของตัวแปร b หรือไม่ ถ้าน้อยกว่าหรือเท่ากับเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่ามากกว่าเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a ⩽ b) มีค่าเป็นจริง คือ 1 (True)
greater than or equal to	⩾	ตรวจสอบว่าค่าของตัวแปร a มีค่ามากกว่าหรือเท่ากับค่าของตัวแปร b หรือไม่ ถ้ามากกว่าหรือเท่ากับเงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่าน้อยกว่าเงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a ⩾ b) มีค่าเป็นเท็จ (False)

6.7.2.1 โอเปอเรเตอร์ที่ใช้ในการตัดสินใจ (Conditional or Ternary Operator)

เป็นตัวดำเนินการตามเงื่อนไขที่มีลักษณะการทำงานคล้ายกับ if-else Statement โดยตัวดำเนินการตามเงื่อนไขนี้จะใช้พื้นที่น้อยกว่า และช่วยในการเขียนคำสั่งเงื่อนไขได้สั้นกว่า

Syntax :



Variable : ตัวแปรผลลัพธ์

Expression1 : เงื่อนไขที่ 1

Expression2 : ถ้าเงื่อนไขที่ 1 เป็นจริง (True) จะทำตามคำสั่งของ Expression2

Expression3 : ถ้าเงื่อนไขที่ 1 เป็นเท็จ (False) จะทำตามคำสั่งของ Expression3

Arduino Tutorial 8 ทดลองการใช้งานตัวดำเนินการเปรียบเทียบ

แล็บการทดลองนี้ เราจะได้เรียนรู้การใช้งานตัวดำเนินการเปรียบเทียบ

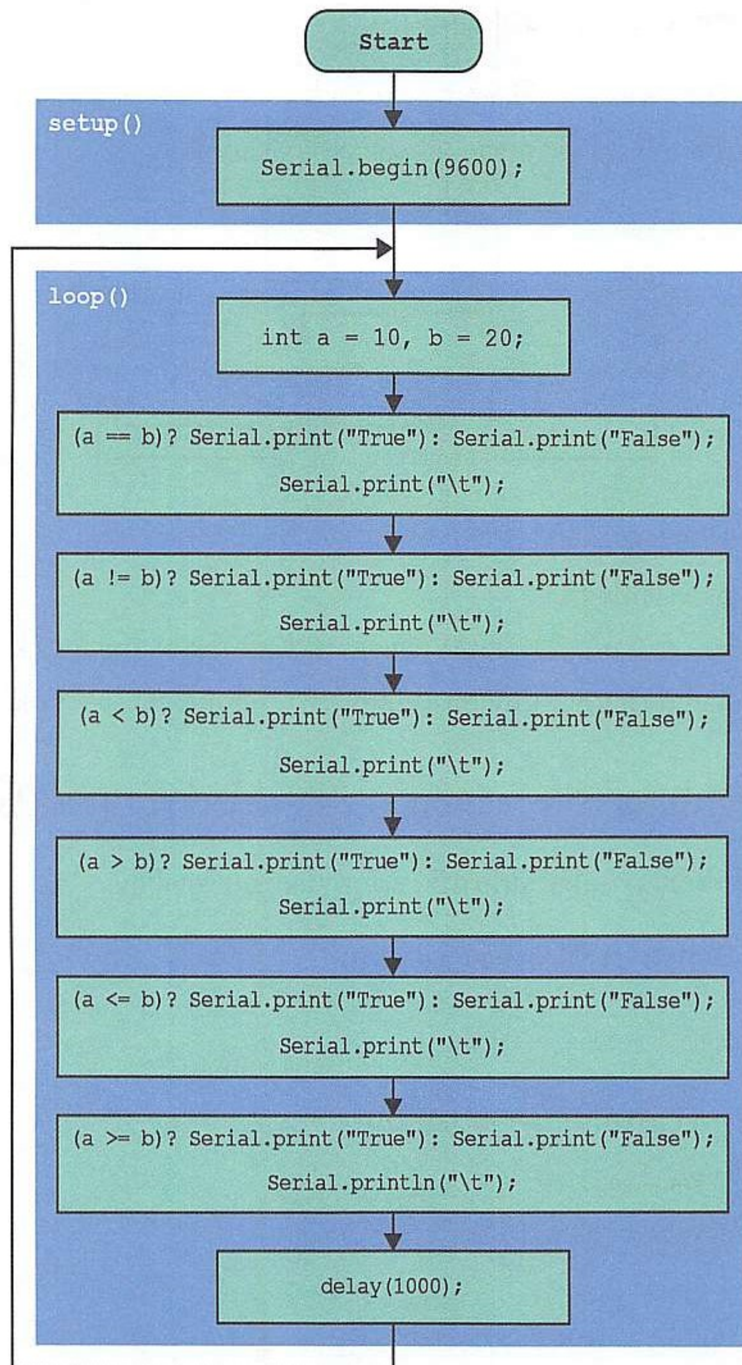
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): Serial.begin()
loop():  Serial.print(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 8



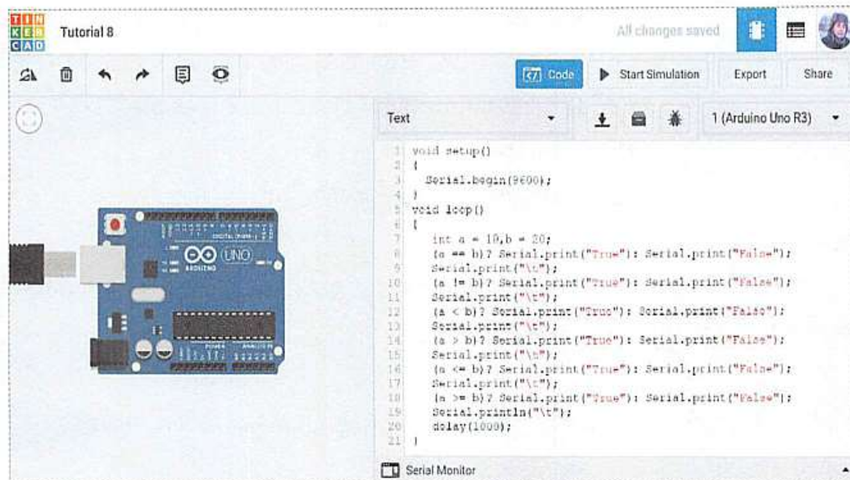
Step 4 : Source Code Arduino_Tutorial_8.ino

```
void setup()              //ฟังก์ชัน setup()
{                         //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);      //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                         //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()               //ฟังก์ชัน loop()
{                         //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  int a = 10,b = 20;
      //กำหนดให้ตัวแปร a เท่ากับ 10 ตัวแปร b เท่ากับ 20 และตัวแปร c เก็บเลขจำนวนเต็ม
  (a == b)? Serial.print("True"): Serial.print("False");

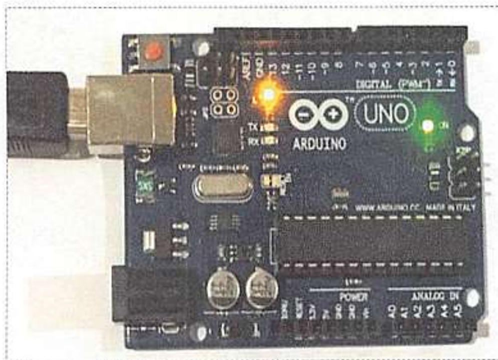
      //สร้างเงื่อนไขถ้า a == b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a != b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a != b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a < b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a < b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a > b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a > b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a <= b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a <= b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a >= b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a >= b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.println("\t");   //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  delay(1000);            //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                         //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

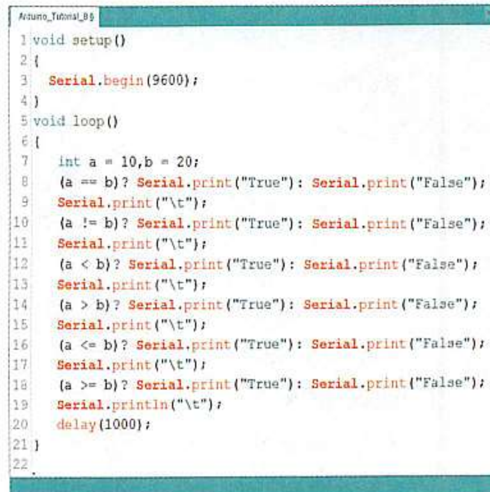
Step 5 : วิธีการทดลอง Arduino Tutorial 8

Tinkercad



Arduino Board





Step 6 : ผลการทดลอง Arduino Tutorial 8

ผลลัพธ์ที่ได้จะแสดงตามเงื่อนไข คือ a = 10, b = 20 เงื่อนไขที่ 1 (a == b) เป็นเท็จ แสดงคำว่า False, เงื่อนไขที่ 2 (a != b) เป็นจริง แสดงคำว่า True, เงื่อนไขที่ 3 (a < b) เป็นจริง แสดงคำว่า True, เงื่อนไขที่ 4 (a > b) เป็นเท็จ แสดงคำว่า False, เงื่อนไขที่ 5 (a <=b) เป็นจริง แสดงคำว่า True, เงื่อนไขที่ 6 (a >=b) เป็นเท็จ แสดงคำว่า False

Tinkercad Output

```
Serial Monitor

False   True    True    False   True    False
False   True    True    False   True    False
False   True    True    False   True    False
```

Arduino Board Output

```
COM3

False   True    True    False   True    False
False   True    True    False   True    False
False   True    True    False   True    False
```

Step 7 : คำถามท้าย Arduino Tutorial 8

1. ทดลองแก้ไขโปรแกรมโดยกำหนดให้ a = 89 และ b = 30 ผลลัพธ์ที่ได้จะเป็นอย่างไร
2. ทดลองเขียนโปรแกรมเปรียบเทียบค่า (a+b) == (b+a) เพื่อแสดงว่าเป็นจริงหรือเท็จ
3. ทดลองเขียนโปรแกรมเปรียบเทียบค่า (a*b) ≤ (b/a) เพื่อแสดงว่าเป็นจริงหรือเท็จ
4. ทดลองเขียนโปรแกรมเปรียบเทียบค่า (a+b/c) > (b/c*a) เพื่อแสดงว่าเป็นจริงหรือเท็จ กำหนดให้ a = 10, b = 20 และ c = 30

6.7.3 ตัวดำเนินการบูลีน (Boolean Operators)

กำหนดให้ตัวแปร A เก็บค่า 10 และตัวแปร B เก็บค่า 20

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	ตัวอย่าง
and	&&	ตัวดำเนินการ AND ถ้าตัวแปร a และตัวแปร b ไม่เป็น 0 ทั้งคู่ เงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้าตัวใดตัวหนึ่งหรือทั้งคู่เป็น 0 เงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a && b) มีค่าเป็นจริง คือ 1 (True)
or	||	ตัวดำเนินการ OR ถ้าตัวแปร a หรือตัวแปร b ตัวใดตัวหนึ่งมีค่าไม่เป็น 0 เงื่อนไขจะเป็นจริง คือ 1 (True) แต่ถ้ามีค่าเป็น 0 ทั้งคู่ เงื่อนไขจะเป็นเท็จ คือ 0 (False)	(a || b) มีค่าเป็นจริง คือ 1 (True)
not	!	ตัวดำเนินการ NOT ใช้เพื่อย้อนกลับสถานะเชิงตรรกะ ถ้าเงื่อนไขเป็นจริง คือ 1 (True) ตัวดำเนินการ NOT จะเป็นเท็จ คือ 0 (False) และถ้าเงื่อนไขเป็นเท็จ คือ 0 (False) ตัวดำเนินการ NOT จะเป็นจริง คือ 1 (True)	!(a && b) มีค่าเป็นเท็จ คือ 0 (False)

Arduino Tutorial 9 ทดลองการใช้งานตัวดำเนินการบูลีน

แล็บการทดลองนี้ เราจะได้เรียนรู้การใช้งานตัวดำเนินการบูลีน

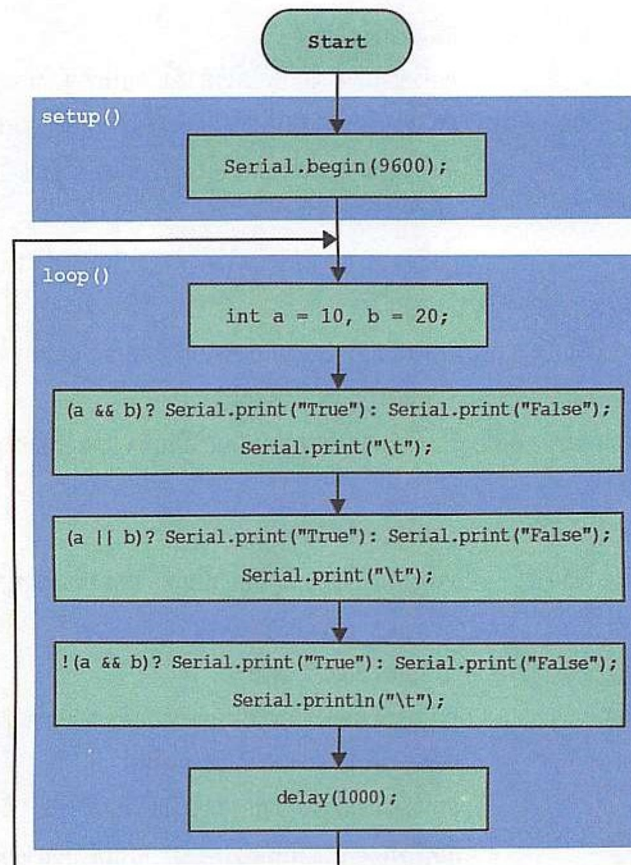
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): Serial.begin()
loop(): Serial.print(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 9



Step 4 : Source Code Arduino_Tutorial_9.ino

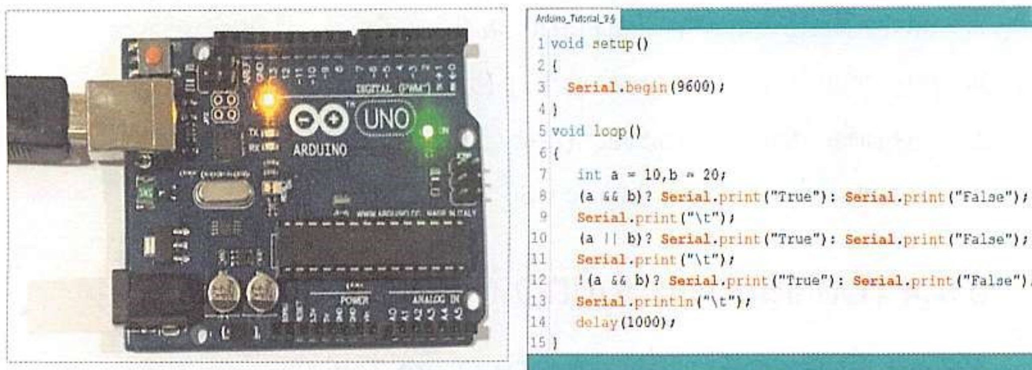
```
void setup()                  //ฟังก์ชัน setup()
{                             //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);          //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                             //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                   //ฟังก์ชัน loop()
{                             //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  int a = 10,b = 20;
      //กำหนดให้ตัวแปร a เท่ากับ 10 ตัวแปร b เท่ากับ 20 และตัวแปร c เก็บเลขจำนวนเต็ม
  (a && b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a && b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  (a || b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า a || b เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.print("\t");
  !(a && b)? Serial.print("True"): Serial.print("False");
      //สร้างเงื่อนไขถ้า !(a && b) เป็นจริง แสดงข้อความ True เป็นเท็จ แสดงข้อความ False
  Serial.println("\t");       //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  delay(1000);                //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                             //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 9

Tinkercad



Arduino Board



Step 6 : ผลการทดลอง Arduino Tutorial 9

ผลลัพธ์ที่ได้จะแสดงตามเงื่อนไข คือ a = 10, b = 20 เงื่อนไขที่ 1 (a && b) เป็นจริง แสดงคำว่า "True," เงื่อนไขที่ 2 (a || b) เป็นจริง แสดงคำว่า "True," เงื่อนไขที่ 3 !(a && b) เป็นเท็จ แสดงคำว่า "False"

Tinkercad Output

```
Serial Monitor
True    True    False
True    True    False
True    True    False
```

Arduino Board Output

```
COM3
True    True    False
True    True    False
True    True    False
```

Step 7 : คำถามท้าย Arduino Tutorial 9

1. ทดลองแก้ไขโปรแกรมโดยกำหนดให้ a = 26 และ b = 0 ผลลัพธ์ที่ได้เป็นอย่างไร
2. ทดลองเขียนโปรแกรมสถานะของ (a+b) && (b+a) เพื่อแสดงว่าเป็นจริงหรือเท็จ
3. ทดลองเขียนโปรแกรมสถานะของ (a*b) || (b/a) เพื่อแสดงว่าเป็นจริงหรือเท็จ
4. ทดลองเขียนโปรแกรมสถานะของ !((a+b/c) && (b/c*a)) เพื่อแสดงว่าเป็นจริงหรือเท็จ โดยกำหนดให้ a = 10, b = 0 และ c = 30

6.7.4 ตัวดำเนินการแบบบิตไวส์ (Bitwise Operators)

6.7.4.1 พื้นฐานตารางค่าความจริง (Truth Table)

ในทางเลขฐานสอง 1 = เป็นจริง (True), 0 = เป็นเท็จ (False) ในตาราง A, B เป็น Input และ Y เป็น Output

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

NOT Truth Table

A	B
0	1
1	0

6.7.4.2 พื้นฐานระบบเลขฐานสอง (Binary Numbers)

ตัวอย่างการแปลงเลขฐานสองเป็นเลขฐานสิบ (Binary to Decimal Conversion)

ระบบเลขฐานสองจะมีค่าน้ำหนักประจำตำแหน่ง 2 ฝั่ง ดังรูป

1. ฝั่งจำนวนเต็ม (Whole Numbers) จะมีค่าน้ำหนัก 2 ตำแหน่งบิต (ตำแหน่งบิตจะเริ่มตั้งแต่ 0, 1, 2, 3, ..., n บิต เรียงจากขวาไปซ้าย)
2. ฝั่งจำนวนทศนิยม (Fractional Numbers) จะมีค่าน้ำหนัก 2 ตำแหน่งบิต (ตำแหน่งบิตจะเริ่มตั้งแต่ -1, -2, -3, -4, ..., -n บิต เรียงจากซ้ายไปขวา)

กำหนดให้เลขฐานสอง คือ 101101.01 ซึ่งตรงกับค่าประจำตำแหน่ง

ฝั่งจำนวนเต็ม คือ $2^5 = 32$, $2^3 = 8$, $2^2 = 4$ และ $2^0 = 1$

ฝั่งจำนวนทศนิยม คือ $2^{-2} = 0.25$

นำค่าน้ำหนักของเลข 1 ทั้งหมดมาบวกกัน 32 + 8 + 4 + 1 + 0.25 จะได้ผลลัพธ์เลขฐานสิบเท่ากับ 45.25

	POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
1	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
	256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
										0.5	0.25	0.125	0.0625	0.03125	0.015625
2				1	0	1	1	0	1 .	0	1				
3				32 +		8 +	4 +		1 +		0.25 = 45.25				

กำหนดให้ตัวแปร A เก็บค่า 60 และตัวแปร B เก็บค่า 30

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	การคำนวณ	ตัวอย่าง
and	&	ตัวดำเนินการบิต AND จะมีกรณีเดียวถ้าค่าบิต (เลขฐานสอง) ของตัวแปร a และตัวแปร b มีค่าบิตเป็น 1 ทั้งคู่ ค่าผลลัพธ์จะเป็น 1 เสมอ นอกนั้นจะมีค่าเป็น 0 (มี 1 ทั้งคู่เป็น 1)	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 b = 0 0 0 1 1 1 1 0 a & b = 0 0 0 1 1 1 0 0	(a & b) ได้ค่าเท่ากับ 28 เลขฐานสอง คือ 0001 1100
or	|	ตัวดำเนินการบิต OR ถ้าค่าบิตของตัวแปร a หรือตัวแปร b มีค่าบิตเป็น 1 ตัวใดตัวหนึ่งหรือทั้งคู่ ค่าผลลัพธ์จะเป็น 1 เสมอ นอกนั้นจะมีค่าเป็น 0 (มี 1 เป็น 1)	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 b = 0 0 0 1 1 1 1 0 A|B = 0 0 1 1 1 1 1 0	(a | b) ได้ค่าเท่ากับ 62 เลขฐานสอง คือ 0011 1110
xor	^	ตัวดำเนินการบิต XOR ถ้าค่าบิตของตัวแปร a และตัวแปร b มีค่าบิตต่างกัน เช่น a เป็น 1 b เป็น 0 หรือ a เป็น 0 b เป็น 1 ค่าผลลัพธ์จะเป็น 1 เสมอ นอกนั้นจะมีค่าเป็น 0 (ต่างกันเป็น 1 เหมือนกันเป็น 0)	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 b = 0 0 0 1 1 1 1 0 a ^ b = 0 0 1 0 0 0 1 0	(a ^ b) ได้ค่าเท่ากับ 34 เลขฐานสอง คือ 0010 0010
not	~	ตัวดำเนินการบิต NOT จะกลับค่าสถานะบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 หรือ ~a = -a-1 ในเลขฐานสิบ	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 ~a = 1 1 0 0 0 0 1 1	(~a) ได้ค่าเท่ากับ -61 เลขฐานสอง คือ 1100 0011 คือ -60-1 = -61

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	การคำนวณ	ตัวอย่าง
shift left	<<	ตัวดำเนินการบิต Shift Left จะเลื่อนบิตไปทางซ้ายตามจำนวนบิตที่ระบุทางด้านขวา เช่น a << 2 ให้เลื่อนบิตไปทางซ้าย 2 ตำแหน่ง และทางขวาสุดให้เติม 0	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 a << 2 = 1 1 1 1 0 0 0 0	a << 2 ได้ค่าเท่ากับ 240 เลขฐานสอง คือ 1111 0000
shift right	>>	ตัวดำเนินการบิต Shift Right จะเลื่อนบิตไปทางขวาตามจำนวนบิตที่ระบุทางด้านขวา เช่น a >> 2 ให้เลื่อนบิตไปทางขวา 2 ตำแหน่ง และทางซ้ายสุดให้เติม 0	2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 128 64 32 16 8 4 2 1 a = 0 0 1 1 1 1 0 0 a >> 2 = 0 0 0 0 1 1 1 1	a >> 2 ได้ค่าเท่ากับ 15 เลขฐานสอง คือ 0000 1111

Arduino Tutorial 10 ทดลองการใช้งานตัวดำเนินการแบบบิต

แล็บการทดลองนี้ เราจะได้เรียนรู้การใช้งานตัวดำเนินการแบบบิต

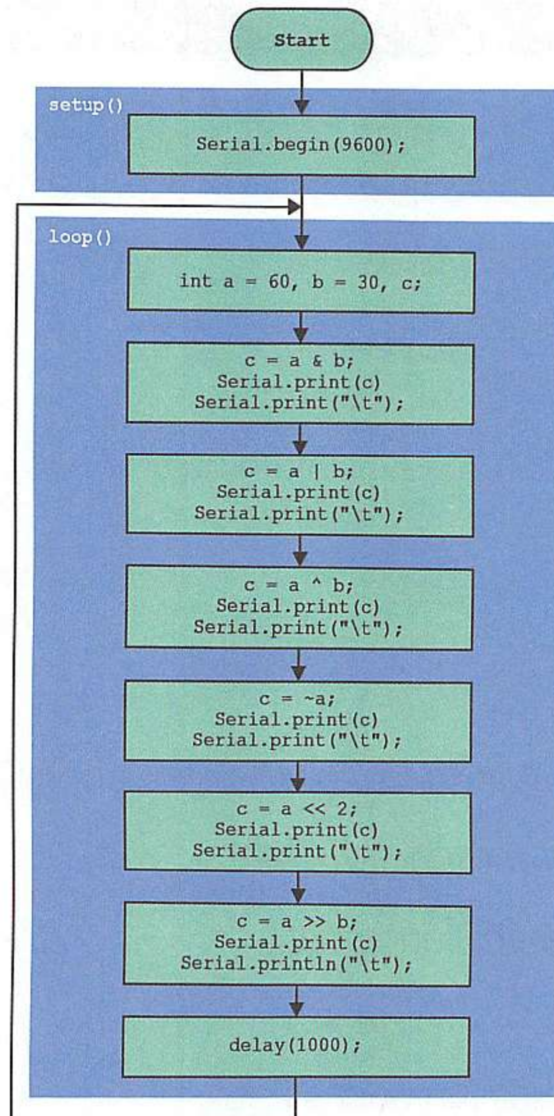
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): Serial.begin()
loop(): Serial.print(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 10

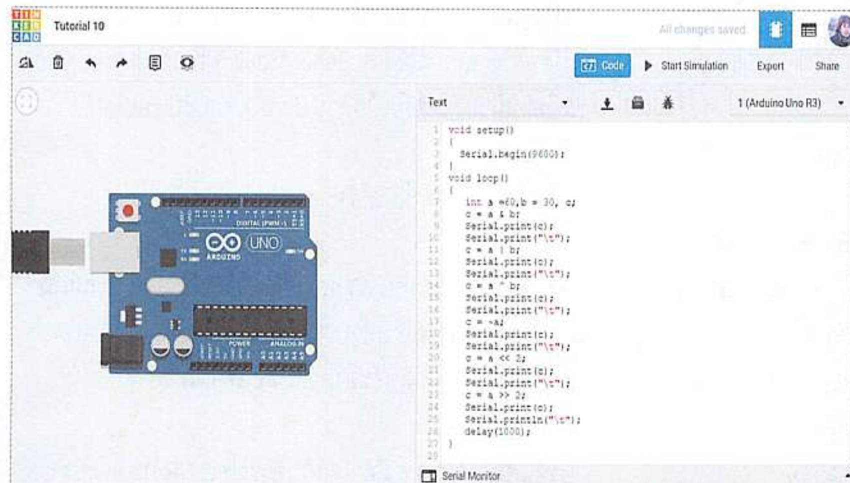


Step 4 : Source Code Arduino_Tutorial_10.ino

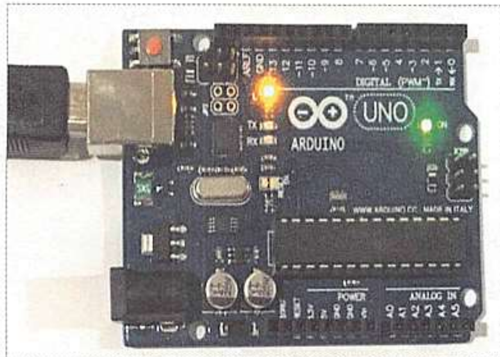
```
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);            //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //ฟังก์ชัน loop()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  int a = 60,b = 30, c;
      //กำหนดให้ตัวแปร a เท่ากับ 60 ตัวแปร b เท่ากับ 30 และตัวแปร c เก็บเลขจำนวนเต็ม
  c = a & b;                    //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a AND กับตัวแปร b
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 28
  Serial.print("\t");           //เว้นวรรค 1 Tab
  c = a | b;                    //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a OR กับตัวแปร b
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 62
  Serial.print("\t");           //เว้นวรรค 1 Tab
  c = a ^ b;                    //กำหนดให้ตัวแปร c มีค่าเท่ากับตัวแปร a XOR กับตัวแปร b
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 34
  Serial.print("\t");           //เว้นวรรค 1 Tab
  c = ~a;                       //กำหนดให้ตัวแปร c มีค่าเท่ากับ NOT ตัวแปร a
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ -61
  Serial.print("\t");           //เว้นวรรค 1 Tab
  c = a << 2;                   //กำหนดให้ตัวแปร c มีค่าเท่ากับ Shift Left ตัวแปร a ไป 2 Bits
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 240
  Serial.print("\t");           //เว้นวรรค 1 Tab
  c = a >> 2;                   //กำหนดให้ตัวแปร c มีค่าเท่ากับ Shift Right ตัวแปร a ไป 2 Bits
  Serial.print(c);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร c เท่ากับ 15
  Serial.println("\t");         //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  delay(1000);                  //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 10

Tinkercad



Arduino Board



Arduino_Tutorial_10 §

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
   int a =60,b = 30, c;
   c = a & b;
   Serial.print(c);
   Serial.print("\t");
   c = a | b;
   Serial.print(c);
   Serial.print("\t");
   c = a ^ b;
   Serial.print(c);
   Serial.print("\t");
   c = ~a;
   Serial.print(c);
   Serial.print("\t");
   c = a << 2;
   Serial.print(c);
   Serial.print("\t");
   c = a >> 2;
   Serial.print(c);
   Serial.println("\t");
   delay(1000);
}
```

Step 6 : ผลการทดลอง Arduino Tutorial 10

ผลลัพธ์ที่ได้จะแสดงตามเงื่อนไข คือ ตั้งค่า a = 60, b = 30 เก็บในตัวแปร c จะได้ c = a & b = 28, c = a | b = 62, c = a ^ b = 34, c = ~a = -61, c = a << 2 = 240 และ c = a >> 2 = 15

Tinkercad Output

```
Serial Monitor

28    62    34    -61    240    15
28    62    34    -61    240    15
28    62    34    -61    240    15
```

Arduino Board Output

```
COM3

28    62    34    -61    240    15
28    62    34    -61    240    15
28    62    34    -61    240    15
```

Step 7 : คำถามท้าย Arduino Tutorial 10

1. ทดลองแก้ไขโปรแกรมโดยกำหนดให้ a = 30 และ b = 89 ผลลัพธ์ที่ได้เป็นอย่างไร
2. ทดลองเขียนโปรแกรมคำนวณค่า c = (a+b) & (b*a) ผลลัพธ์ที่ได้เป็นอย่างไร
3. ทดลองเขียนโปรแกรมคำนวณค่า c = (a^b) | (b & (a<<3)) ผลลัพธ์ที่ได้เป็นอย่างไร
4. ทดลองเขียนโปรแกรมคำนวณค่า d = ((a >> (c / a)) & (b^(c << 3))) กำหนดให้ a = 10, b = 20 และ c = 30 ผลลัพธ์ที่ได้เป็นอย่างไร

6.7.5 ตัวดำเนินการผสม (Compound Operators)

กำหนดให้ตัวแปรเริ่มต้น A เก็บค่า 69 และ B เก็บค่า 30 (เนื่องจากเก็บค่าอัพเดตใหม่เข้าตัวแปรเดิม จึงสามารถนำมาใช้ต่อได้)

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	ตัวอย่าง
increment	++	ตัวดำเนินการเพิ่มค่า เพิ่มค่าจำนวนเต็มบวก 1	a++ = 69++ จะได้ค่า a ใหม่ คือ 70
decrement	--	ตัวดำเนินการลดค่า ลดค่าจำนวนเต็มลบ 1	a-- = 70-- จะได้ค่า a ใหม่ คือ 69
compound addition	+=	ตัวดำเนินการมอบหมายและเพิ่มค่ามาเก็บที่ตัวเอง เช่น a += b จะมีค่าเท่ากับ a = a + b	a += b มีค่าเท่ากับ a = a + b = 69 + 30 จะได้ค่า a ใหม่ คือ 99
compound subtraction	-=	ตัวดำเนินการมอบหมายและลดค่ามาเก็บที่ตัวเอง เช่น a -= b จะมีค่าเท่ากับ a = a - b	a -= b มีค่าเท่ากับ a = a – b = 99 – 30 จะได้ค่า a ใหม่ คือ 69
compound multiplication	*=	ตัวดำเนินการมอบหมายและคูณค่ามาเก็บที่ตัวเอง เช่น a *= b จะมีค่าเท่ากับ a = a * b	a *= b มีค่าเท่ากับ a = a * b = 69 * 30 จะได้ค่า a ใหม่ คือ 2070
compound division	/=	ตัวดำเนินการมอบหมายและหารค่ามาเก็บที่ตัวเอง เช่น a /= b จะมีค่าเท่ากับ a = a / b	a /= b มีค่าเท่ากับ a = a / b = 2070 / 30 จะได้ค่า a ใหม่ คือ 69
compound modulo	%=	ตัวดำเนินการมอบหมายและหารเอาค่าเศษมาเก็บที่ตัวเอง เช่น a %= b จะมีค่าเท่ากับ a = a % b	a %= b มีค่าเท่ากับ a = a % b = 69 % 30 จะได้ค่า a ใหม่ คือ 9

ชื่อโอเปอเรเตอร์	สัญลักษณ์	รายละเอียด	ตัวอย่าง
compound bitwise and	&=	ตัวดำเนินการบิตและ AND ค่ามาเก็บที่ตัวเอง เช่น a &= b จะมีค่าเท่ากับ a = a & b	a &= b มีค่าเท่ากับ a = a & b = 9 & 30 จะได้ค่า a ใหม่ คือ 8 27 26 25 24 23 22 21 20 128 64 32 16 8 4 2 1 a = 0 0 0 0 1 0 0 1 b = 0 0 0 1 1 1 1 0 a & b = 0 0 0 0 1 0 0 0
compound bitwise or	|=	ตัวดำเนินการบิตและ OR ค่ามาเก็บที่ตัวเอง เช่น a |= b จะมีค่าเท่ากับ a = a | b	a |= b มีค่าเท่ากับ a = a | b = 8 | 30 จะได้ค่า a ใหม่ คือ 30 27 26 25 24 23 22 21 20 128 64 32 16 8 4 2 1 a = 0 0 0 0 1 0 0 0 b = 0 0 0 1 1 1 1 0 a | b = 0 0 0 1 1 1 1 0

Arduino Tutorial 11 ทดลองการใช้งานตัวดำเนินการผสม

ใน Arduino Tutorial นี้ เราจะได้เรียนรู้การใช้งานตัวดำเนินการผสม

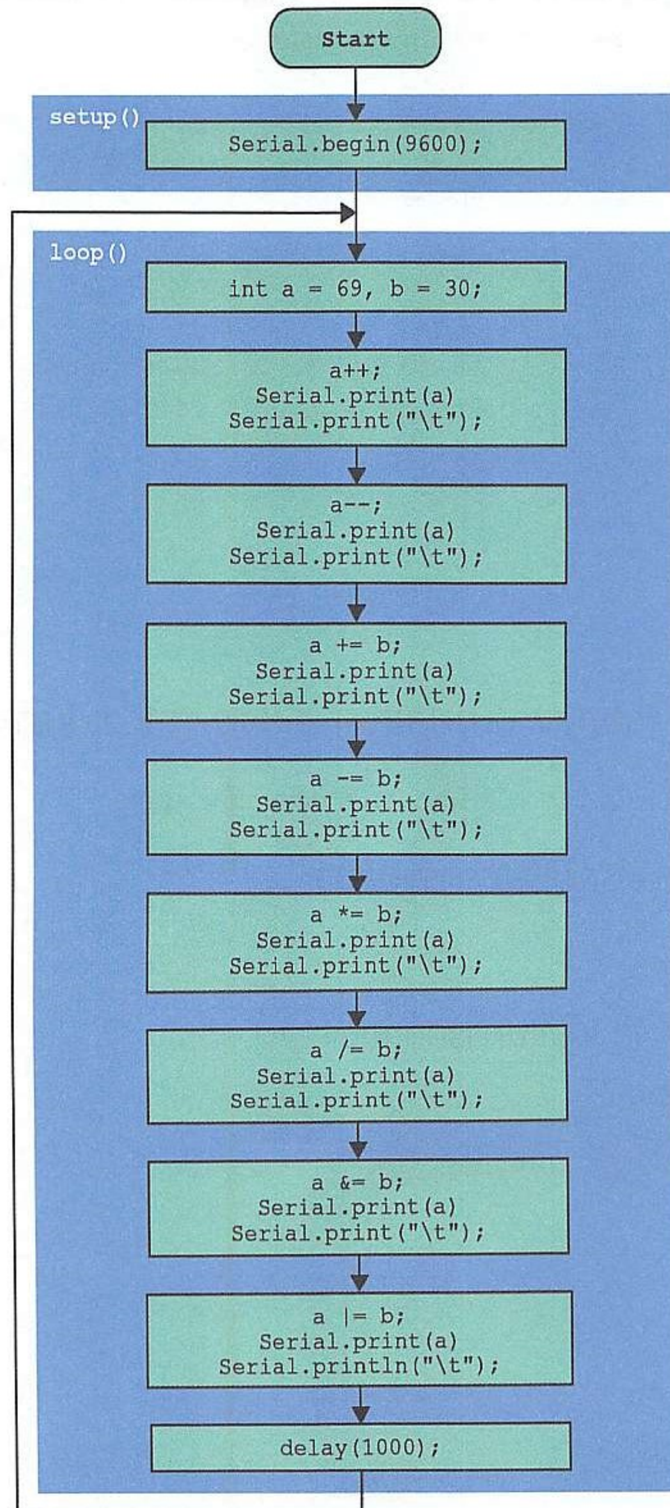
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

บอร์ด Arduino 1 บอร์ด

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): Serial.begin()
loop(): Serial.print(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 11

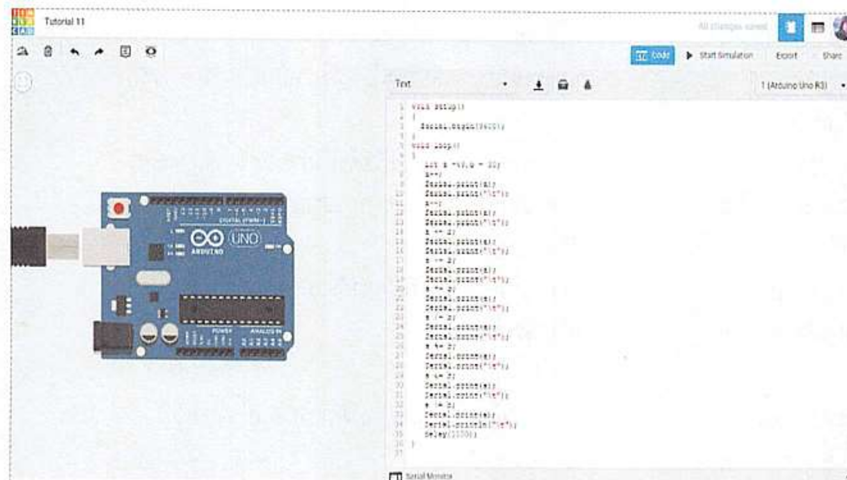


Step 4 : Source Code Arduino_Tutorial_11.ino

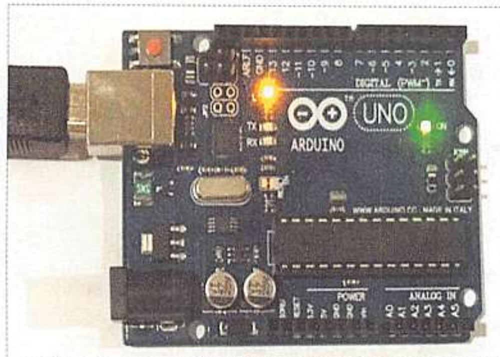
```
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 Serial.begin(9600);            //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //ฟังก์ชัน loop()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  int a = 69,b = 30;            //กำหนดให้ตัวแปร a เท่ากับ 69 และตัวแปร b เท่ากับ 30
  a++;                          //เพิ่มค่า a บวก 1
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 70
  Serial.print("\t");           //เว้นวรรค 1 Tab
  a--;                          //ลดค่า a ลบ 1
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 69
  Serial.print("\t");           //เว้นวรรค 1 Tab
  a += b;                       //นำค่า a บวก b เก็บที่ตัวแปร a (a = a + b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 99
  Serial.print("\t");           //เว้นวรรค 1 Tab
  a -= b;                       //นำค่า a ลบ b เก็บที่ตัวแปร a (a = a - b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 69
  Serial.print("\t");           //เว้นวรรค 1 Tab
  a *= b;                       //นำค่า a คูณ b เก็บที่ตัวแปร a (a = a * b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 2070
  Serial.print("\t");           //เว้นวรรค 1 Tab
  a /= b;                       //นำค่า a หาร b เก็บที่ตัวแปร a (a = a / b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 69
  Serial.print("\t");           //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  a %= b;                       //นำค่า a มอด b เก็บที่ตัวแปร a (a = a % b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 9
  Serial.print("\t");           //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  a &= b;                       //นำค่า a AND b เก็บที่ตัวแปร a (a = a & b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 8
  Serial.print("\t");           //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  a |= b;                       //นำค่า a OR b เก็บที่ตัวแปร a (a = a | b)
  Serial.print(a);              //แสดงค่าข้อมูลผลลัพธ์ของตัวแปร a เท่ากับ 30
  Serial.println("\t");         //เว้นวรรค 1 Tab และขึ้นบรรทัดใหม่
  delay(1000);                  //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

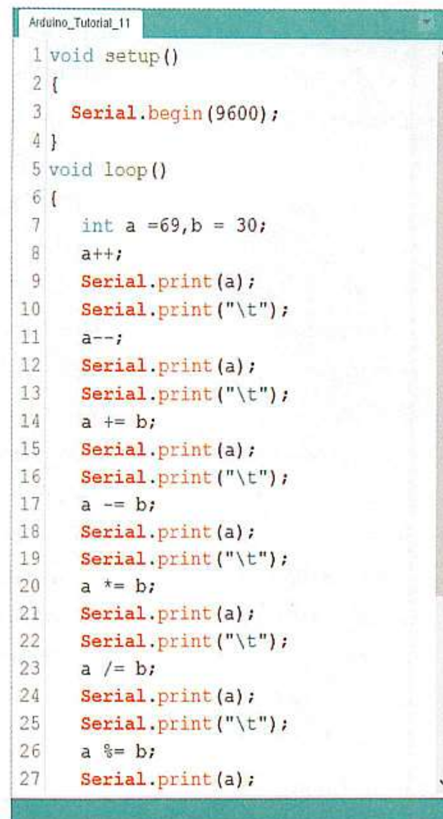
Step 5 : วิธีการทดลอง Arduino Tutorial 11

Tinkercad



Arduino Board





Step 6 : ผลการทดลอง Arduino Tutorial 11

ผลลัพธ์ที่ได้จะแสดงตามเงื่อนไข คือ ตั้งค่า a = 69, b = 30 เก็บในตัวแปร c จะได้ a++ = 70, a- - = 69, a += b = 99, a -= b = 69, a *= b = 2070, a /= b = 69, a %= b = 9, a &= b = 8 และ a |= b = 30

Tinkercad Output

Serial Monitor

```
70    69    99    69    2070    69    9    8    30
70    69    99    69    2070    69    9    8    30
70    69    99    69    2070    69    9    8    30
```

Arduino Board Output

COM3

Send

```
70    69    99    69    2070    69    9    8    30
70    69    99    69    2070    69    9    8    30
70    69    99    69    2070    69    9    8    30
```

Step 7 : คำถามท้าย Arduino Tutorial 11

1. ทดลองแก้ไขโปรแกรมโดยกำหนดให้ a = 30 และ b = 69 ผลลัพธ์ที่ได้เป็นอย่างไร
2. ทดลองเขียนโปรแกรมคำนวณค่า a += (a*b/20) ผลลัพธ์ที่ได้เป็นอย่างไร
3. ทดลองเขียนโปรแกรมคำนวณค่า b *= (a-b*(26%a)) ผลลัพธ์ที่ได้เป็นอย่างไร
4. ทดลองเขียนโปรแกรมคำนวณค่า c *= (a/b%(a+60)) กำหนดให้ a = 10, b = 20 และ c = 30 ผลลัพธ์ที่ได้เป็นอย่างไร

6.8 การใช้คำสั่งทำซ้ำ (for loop/while loop/do while)

ในหัวข้อนี้เราจะเพิ่มความซับซ้อนในการเขียนโปรแกรม Arduino เพื่อให้โปรแกรมทำงานซ้ำอัตโนมัติด้วยคำสั่งลูป (Loop) แบบต่างๆ ได้แก่ for loop, while loop และ do while โดยเราจะใช้วงจรเดียวกับ Arduino Tutorial 5 และเรียกใช้คำสั่ง digitalWrite เพื่อช่วยในการแสดงผลค่าของ LED พร้อมกับแสดงข้อความบนหน้าจอ Serial Monitor

จากโจทย์ในการทดลอง Arduino Tutorial 12–15 เราจะควบคุมจำนวนรอบการกะพริบของหลอดไฟ LED ให้สามารถกะพริบติดและดับได้ตามจำนวนที่เรากำหนด โดยได้นำ Arduino Tutorial 5 มาปรับแต่งใหม่ด้วยการใช้คำสั่งแบบทั่วไป และคำสั่งแบบวนซ้ำหรือวนลูปในรูปแบบต่างๆ มาช่วยควบคุมจำนวนรอบการกะพริบ โดยในที่นี้จะกำหนดให้หลอดไฟ LED กะพริบติดและดับจำนวน 5 รอบ

Arduino Tutorial 12 ทดลองการใช้งานคำสั่งแบบทั่วไปในการควบคุมไฟกะพริบ

โดยเราจะใช้คำสั่งแบบเดียวกับ Arduino Tutorial 5 และใช้คำสั่ง digitalWrite แสดงผลค่าของ LED ให้กะพริบติดและดับจำนวน 5 รอบ พร้อมกับแสดงข้อความบนหน้าจอ Serial Monitor ซึ่งจะเป็นแบบกำหนดเอง (Manual) ไม่ใช่แบบอัตโนมัติ (Automatic)

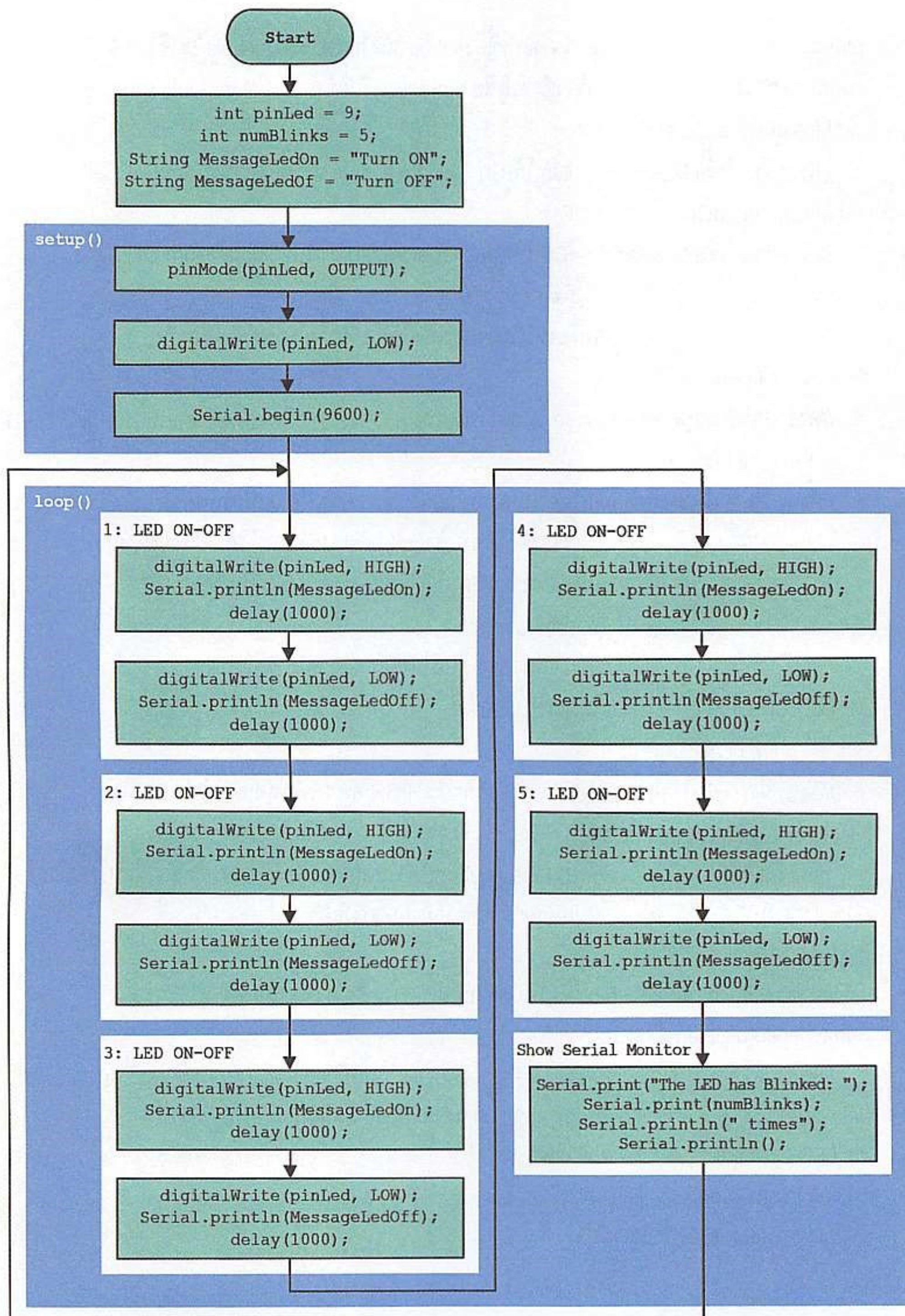
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino	1	บอร์ด
2. หลอดไฟ LED	1	ดวง
3. ตัวต้านทาน 220 โอห์ม	1	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay()
```

Step 3 : Flowchart Arduino Tutorial 12



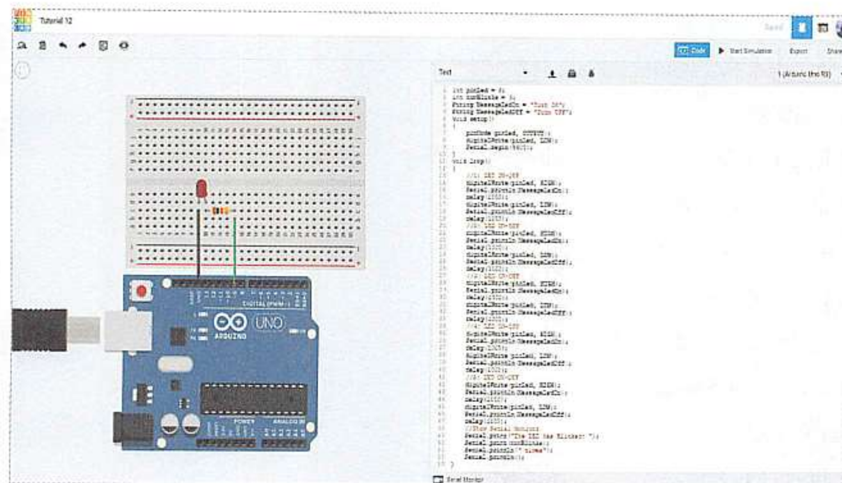
Step 4 : Source Code Arduino_Tutorial_12.ino

```
int pinLed = 9;              //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
int numBlinks = 5;           //สร้างตัวแปรชื่อ numBlinks ให้เป็น Global Variable เก็บค่า 5
String MessageLedOn = "Turn ON";
     //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ "Turn ON"
String MessageLedOff = "Turn OFF";
     //สร้างตัวแปรชื่อ MessageLedOff ให้เป็น Global Variable เก็บข้อความ "Turn OFF"
void setup()                 //ฟังก์ชัน setup()
{                            //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(pinLed, OUTPUT);
     //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(pinLed, LOW);
     //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);        //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                            //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                  //ฟังก์ชัน loop()
{                            //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
  //1: LED ON-OFF            //ไฟกะพริบรอบที่ 1
  digitalWrite(pinLed, HIGH);
     //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println(MessageLedOn);
     //แสดงผลข้อความที่เก็บในตัวแปร MessageLedOn บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);               //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
     //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println(MessageLedOff);
     //แสดงผลข้อความที่เก็บในตัวแปร MessageLedOFF บนหน้าจอ Serial Monitor และขึ้นบรรทัดใหม่
  delay(1000);               //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  //2: LED ON-OFF            //ไฟกะพริบรอบที่ 2
  digitalWrite(pinLed, HIGH);
  Serial.println(MessageLedOn);
```

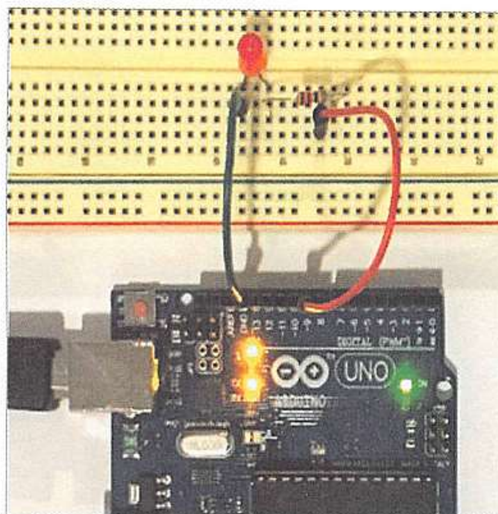
```
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    //3: LED ON-OFF              //ไฟกะพริบรอบที่ 3
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    //4: LED ON-OFF              //ไฟกะพริบรอบที่ 4
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    //5: LED ON-OFF              //ไฟกะพริบรอบที่ 5
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    //Show Serial Monitor
    Serial.print("The LED has Blinked: "); //แสดงข้อความ The LED has Blinked:
    Serial.print(numBlinks);     //แสดงค่าที่เก็บในตัวแปร numBlinks เพื่อบอกว่าครบจำนวน 5 รอบแล้ว
    Serial.println(" times");    //แสดงข้อความ times
    Serial.println();            //ขึ้นบรรทัดใหม่หรือใช้คำสั่ง Serial.print("\n") เพื่อขึ้นบรรทัดใหม่ได้เช่นกัน
}
```

Step 5 : วิธีการทดลอง Arduino Tutorial 12

Tinkercad



Arduino Board



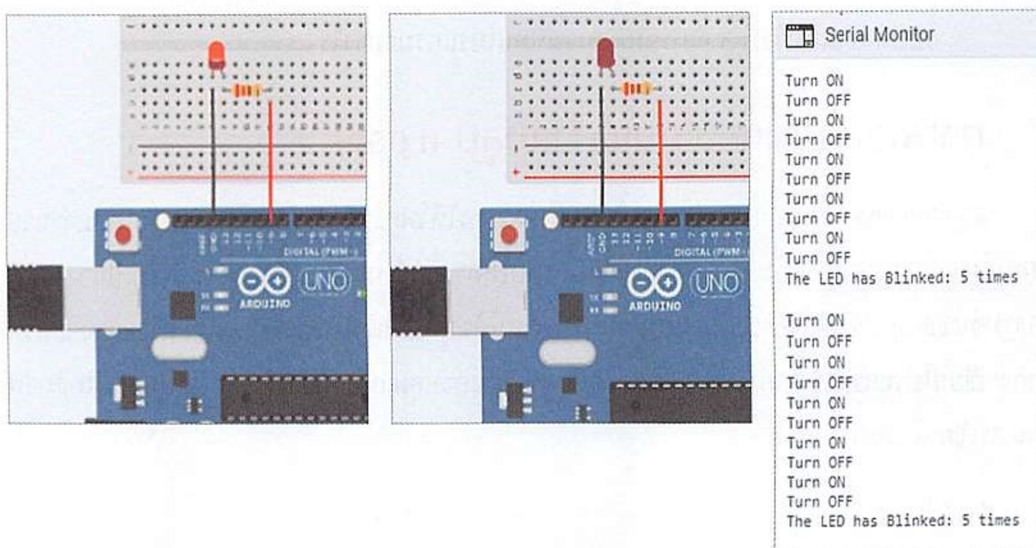
Arduino_Tutorial_12

```
int pinLed = 9;
int numBlinks = 5;
String MessageLedOn = "Turn ON";
String MessageLedOff = "Turn OFF";
void setup()
{
    pinMode(pinLed, OUTPUT);
    digitalWrite(pinLed, LOW);
    Serial.begin(9600);
}
void loop()
{
    //1: LED ON-OFF
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    //2: LED ON-OFF
```

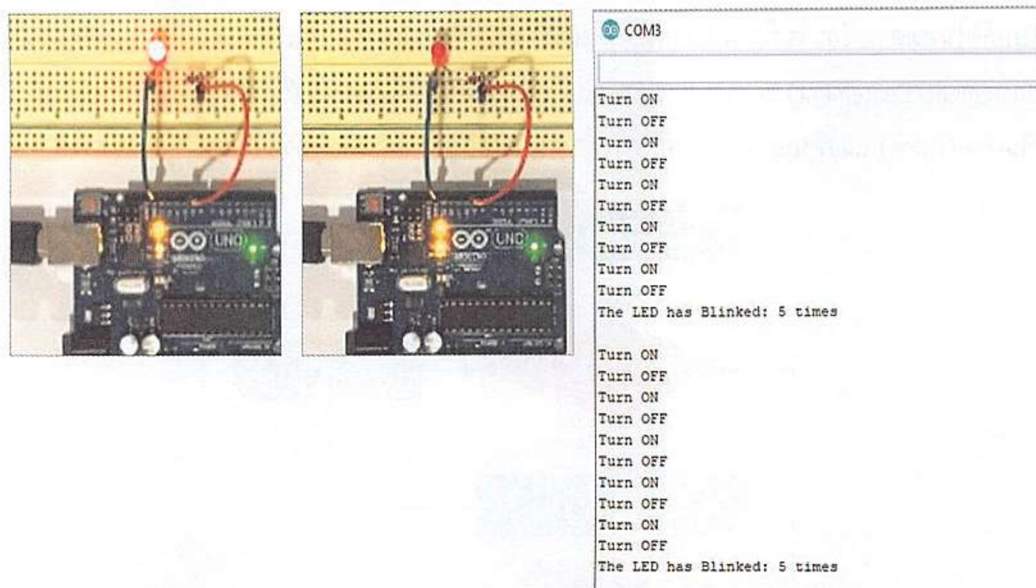
Step 6 : ผลการทดลอง Arduino Tutorial 12

ผลลัพธ์ที่ได้จะแสดงการกะพริบไฟติดและดับจำนวน 5 รอบ แล้วจะแสดงข้อความว่ากะพริบไฟครบ 5 รอบแล้ว และจะทำซ้ำเดิมไปเรื่อยๆ สังเกตจำนวนบรรทัดของโค้ดที่ต้องพิมพ์เพื่อนำไปเปรียบเทียบกับแล็บต่อไป

Tinkercad Output



Arduino Board Output



Step 7 : คำถามท้าย Arduino Tutorial 12

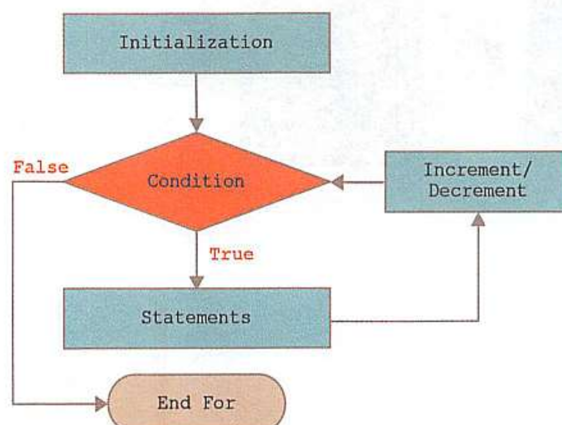
1. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบทุกๆ 3 รอบ จะเขียนโปรแกรมอย่างไร
2. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบทุกๆ 8 รอบ จะเขียนโปรแกรมอย่างไร
3. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบครั้งที่ 1 คือ รอบที่ 2, ครั้งที่ 2 คือ รอบที่ 4 และครั้งที่ 3 คือ รอบที่ 6 จะเขียนโปรแกรมอย่างไร
4. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบครั้งที่ 1 คือ รอบที่ 3, ครั้งที่ 2 คือ รอบที่ 6 และครั้งที่ 3 คือ รอบที่ 9 จะเขียนโปรแกรมอย่างไร

การเขียนโปรแกรมโดยใช้คำสั่งลูป (Loop)

จาก Arduino Tutorial 12 จะสังเกตเห็นว่า ถ้าต้องการให้ไฟกะพริบติดและดับแสดงข้อความจำนวนรอบที่มากขึ้น เราจะต้องเพิ่มโค้ดแสดงการติดและดับให้ตรงกับจำนวนรอบที่ต้องการ ซึ่งถ้าเป็นวงจรที่มีความซับซ้อนมากขึ้นอาจจะไม่เหมาะสม เพราะต้องแก้โค้ดทุกครั้งแบบ Manual ดังนั้น เพื่อความสะดวก เราจะเขียนโปรแกรมให้มันทำงานซ้ำอัตโนมัติโดยใช้คำสั่งควบคุมแบบวนซ้ำ ซึ่งจะช่วยเพิ่มประสิทธิภาพในการเขียนโปรแกรม ดังนี้

for loop

คำสั่ง for loop ใช้เพื่อดำเนินการทำซ้ำคำสั่งเดิมไปเรื่อยๆ ตามจำนวนรอบการทำงานที่กำหนดแบบอัตโนมัติตามเงื่อนไขที่เป็นจริง (True) โดยเมื่อทำคำสั่งในแต่ละครั้งเสร็จแล้ว จะเพิ่มค่าหรือลดค่า (Increment/Decrement) ของรอบการทำซ้ำไปเรื่อยๆ จนกว่าจะไม่ตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขเป็นเท็จ (False) แล้วจึงออกจากวงรอบการทำงาน หรือออกจากลูป for ดังแสดงในรูปโฟลว์ชาร์ต



Syntax :

```
for(initialization; condition; increment/decrement)
    //คำสั่ง for (กำหนดค่าเริ่มต้น; กำหนดเงื่อนไขถ้าจริงจะทำ; เพิ่มหรือลดค่าเริ่มต้นให้วน Loop)
{
    Statements or Body of the loop;     //คำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง
}
```

Parameters :

initialization : กำหนดค่าเริ่มต้นครั้งแรก

condition : กำหนดเงื่อนไข โดยแต่ละครั้งที่ผ่านลูปเงื่อนไขจะถูกทดสอบ ถ้าเป็นจริงจะทำตามคำสั่ง และดำเนินการเพิ่มค่าหรือลดค่าเริ่มต้น ตราบใดที่เงื่อนไขเป็นจริงก็จะทำคำสั่งเดิมวนลูปไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จ ก็จะถือว่าสิ้นสุดการวนลูปคำสั่ง

increment/decrement : ดำเนินการเพิ่มค่าหรือลดค่าทุกครั้ง เมื่อผ่านการทำคำสั่งกรณีเงื่อนไขเป็นจริง

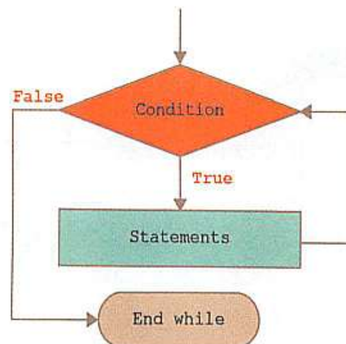
ตัวอย่างคำสั่งลูปแบบ Increment

```
for(i=0; i<5; i++) {
print("Hello Arduino");
}
```

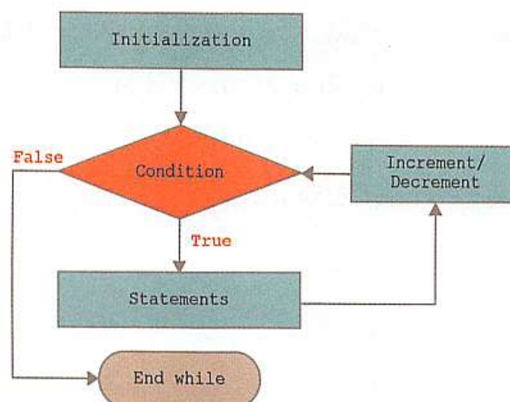
ผลลัพธ์จะแสดงข้อความ Hello Arduino จำนวน 5 รอบ สังเกตว่าการใช้คำสั่งลูป for ช่วยประหยัดบรรทัดคำสั่งได้มากเมื่อเทียบกับการเขียนโปรแกรมแบบ Manual

while loop

คำสั่ง while เป็นคำสั่งที่ทำงานได้เช่นเดียวกับคำสั่ง for แต่เขียน Syntax ต่างกัน โดยคำสั่ง while จะมีหรือไม่มีการกำหนดค่าเริ่มต้น (Initialization) และเพิ่มค่าหรือลดค่า (Increment/Decrement) ก็ได้ขึ้นอยู่กับเงื่อนไขที่กำหนด สามารถดำเนินการทำซ้ำคำสั่งเดิมไปเรื่อยๆ ตามจำนวนรอบการทำงานที่กำหนดแบบอัตโนมัติตามเงื่อนไขที่เป็นจริง โดยเมื่อทำคำสั่งในแต่ละครั้งเสร็จแล้วจะเพิ่มค่าหรือลดค่าของรอบการทำซ้ำไปเรื่อยๆ จนกว่าจะไม่ตรงกับเงื่อนไขที่กำหนด แล้วจึงออกจากลูปการทำงาน



หรือ



Syntax :

while (condition)

//คำสั่ง while (กำหนดเงื่อนไขถ้าจริงจะทำ หรือใส่ค่า True เป็นจริงเสมอจะทำตลอด แต่ถ้าเป็นเท็จจะออกนอก loop)

{

Statements or Body of the loop; //คำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง

}

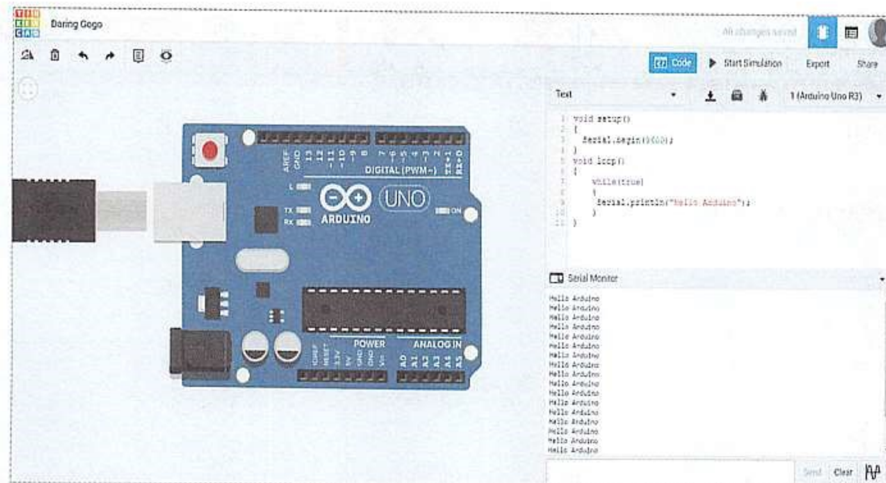
Parameters :

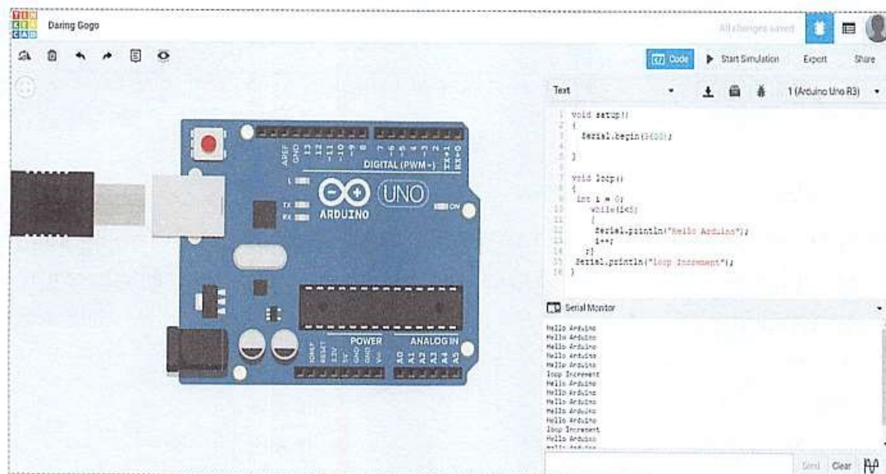
condition : กำหนดเงื่อนไข โดยแต่ละครั้งที่ผ่านลูปเงื่อนไขจะถูกทดสอบ ถ้าเป็นจริงจะทำตามคำสั่งและทำคำสั่งเดิมวนลูปอีกครั้ง ถ้าเป็นเท็จจะสิ้นสุดการวนลูปคำสั่ง

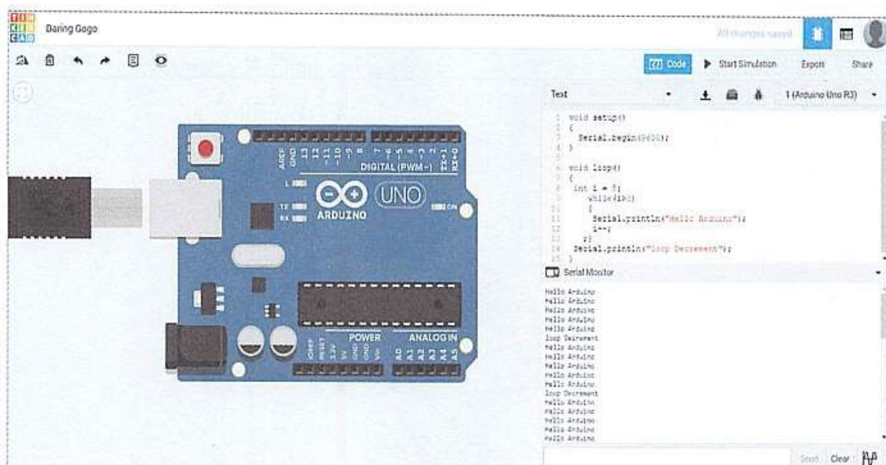
เช่น ถ้ามีเฉพาะ Condition หรือถ้ามีการกำหนด Initialization และ Increment/Decrement จะได้

แบบมีเฉพาะ Condition	แบบ Increment	แบบ Decrement
void setup() { Serial.begin(9600); } void loop() { while(true) { Serial.println("Hello Arduino"); } }	void setup() { Serial.begin(9600); } void loop() { int i = 0; while(i<5) { Serial.println("Hello Arduino"); i++; ;} Serial.println("loop Increment"); }	void setup() { Serial.begin(9600); } void loop() { int i = 5; while(i>0) { Serial.println("Hello Arduino"); i--; ;} Serial.println("loop Decrement"); }
ผลลัพธ์ แสดงข้อความ Hello Arduino ซ้ำๆ ไม่รู้จบ Hello Arduino Hello Arduino Hello Arduino Hello Arduino Hello Arduino (ซ้ำไปเรื่อยๆ)	**ผลลัพธ์** แสดงข้อความ Hello Arduino จำนวน 5 ครั้ง ในลูป while และออกจากลูปด้วยข้อความ loop Increment/Decrement แล้วกลับเข้าสู่ฟังก์ชัน loop เพื่อทำซ้ำลูป while Hello Arduino Hello Arduino Hello Arduino Hello Arduino Hello Arduino loop Increment/Decrement (ซ้ำไปเรื่อยๆ)	

ทดลองเขียนโปรแกรมและรันดูผลลัพธ์ใน Thinkercad

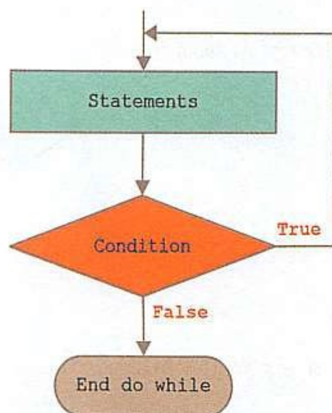




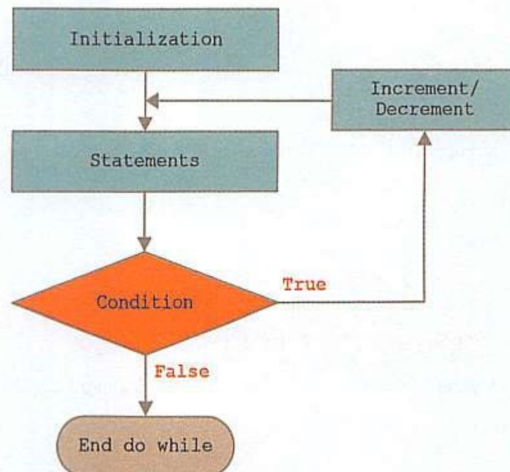


do while loop

คำสั่ง do while เป็นคำสั่งวนซ้ำที่จะทำตามคำสั่งก่อนครั้งแรกแล้วจึงมาเช็คเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะวนไปทำคำสั่งนั้นต่อ



หรือ



Syntax :

```
do        //ทำคำสั่งก่อน
{
  Statements or Body of the loop;    //คำสั่งที่ทำในครั้งแรกก่อนเข้าลูป
} while (condition);

        //คำสั่ง while (กำหนดเงื่อนไขถ้าจริงจะทำต่อ ถ้าเป็นเท็จจะออกจาก loop)
```

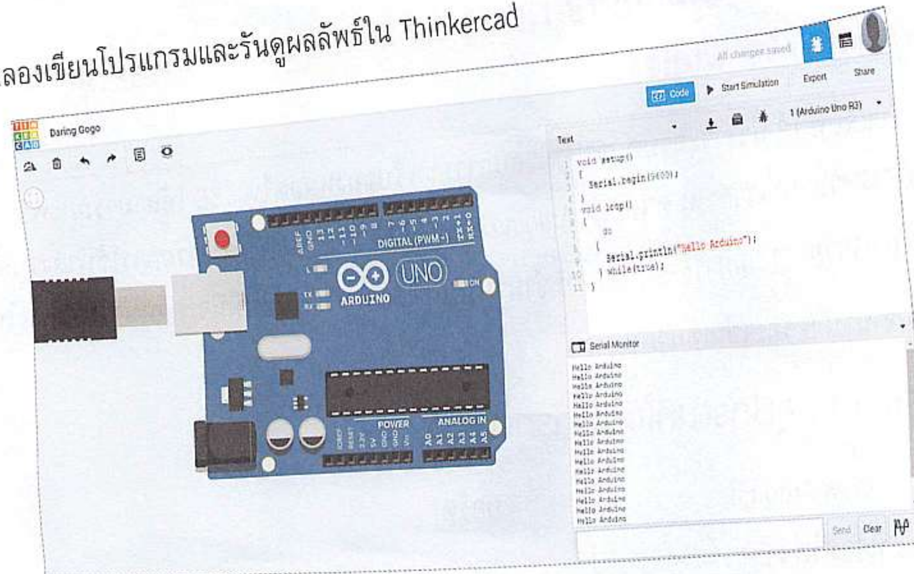
Parameters :

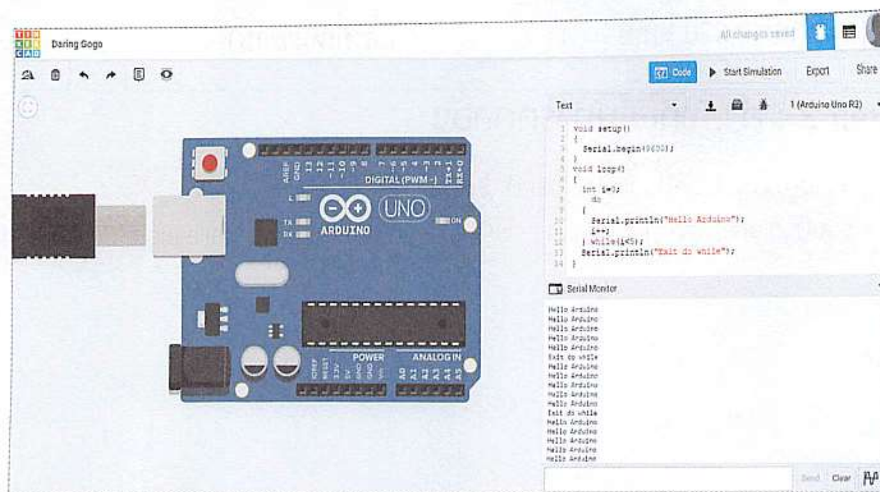
condition : กำหนดเงื่อนไข โดยแต่ละครั้งที่ผ่านลูปเงื่อนไขจะถูกทดสอบ ถ้าเป็นจริงจะวนกลับไปทำตามคำสั่ง และกลับมาเช็คเงื่อนไขไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จ คือ สิ้นสุดการวนลูป

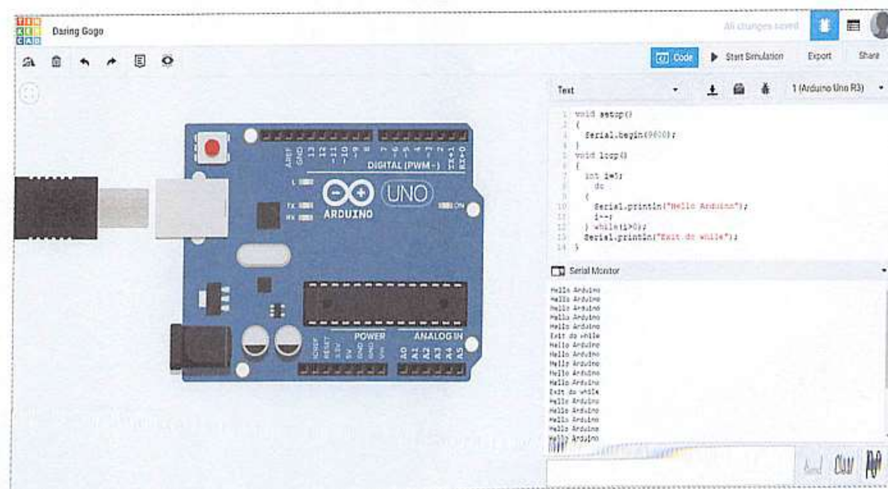
เช่น ถ้ามีเฉพาะ Condition หรือถ้ามีการกำหนด Initialization และ Increment/Decrement จะได้

แบบมีเฉพาะ Condition	แบบ Increment	แบบ Decrement
void setup() { Serial.begin(9600); } void loop() { do { Serial.println("Hello Arduino"); } while(true); }	void setup() { Serial.begin(9600); } void loop() { int i=0; do { Serial.println("Hello Arduino"); i++; } while(i<5); Serial.println("Exit do while"); }	void setup() { Serial.begin(9600); } void loop() { int i=5; do { Serial.println("Hello Arduino"); i--; } while(i>0); Serial.println("Exit do while"); }
ผลลัพธ์ แสดงข้อความซ้ำๆ ไม่รู้จบ Hello Arduino Hello Arduino Hello Arduino (ซ้ำไปเรื่อยๆ)	**ผลลัพธ์** แสดงข้อความ Hello 5 ครั้ง และเมื่อเงื่อนไขเป็นเท็จ จะออกจากลูป do while และพิมพ์ Exit do while แล้วกลับเข้าฟังก์ชัน loop() Hello Arduino Hello Arduino Hello Arduino Hello Arduino Hello Arduino Exit do while (ซ้ำไปเรื่อยๆ)	

ทดลองเขียนโปรแกรมและรันดูผลลัพธ์ใน Thinkercad







Arduino Tutorial 13 ทดลองการใช้งานคำสั่งวนซ้ำแบบ for/while/do while

ในแล็บที่ 13 นี้ เราจะควบคุมจำนวนรอบการกะพริบของหลอดไฟ LED ให้สามารถกะพริบติดและดับได้ตามจำนวนที่เรากำหนด โดยนำ Arduino Tutorial 12 มาปรับแต่งใหม่ด้วยการใช้คำสั่งวนซ้ำแบบ for loop มาช่วยควบคุมการแสดงข้อความจำนวนรอบการกะพริบ โดยในที่นี้จะกำหนดให้หลอดไฟ LED กะพริบจำนวน 5 รอบเช่นเดียวกัน

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

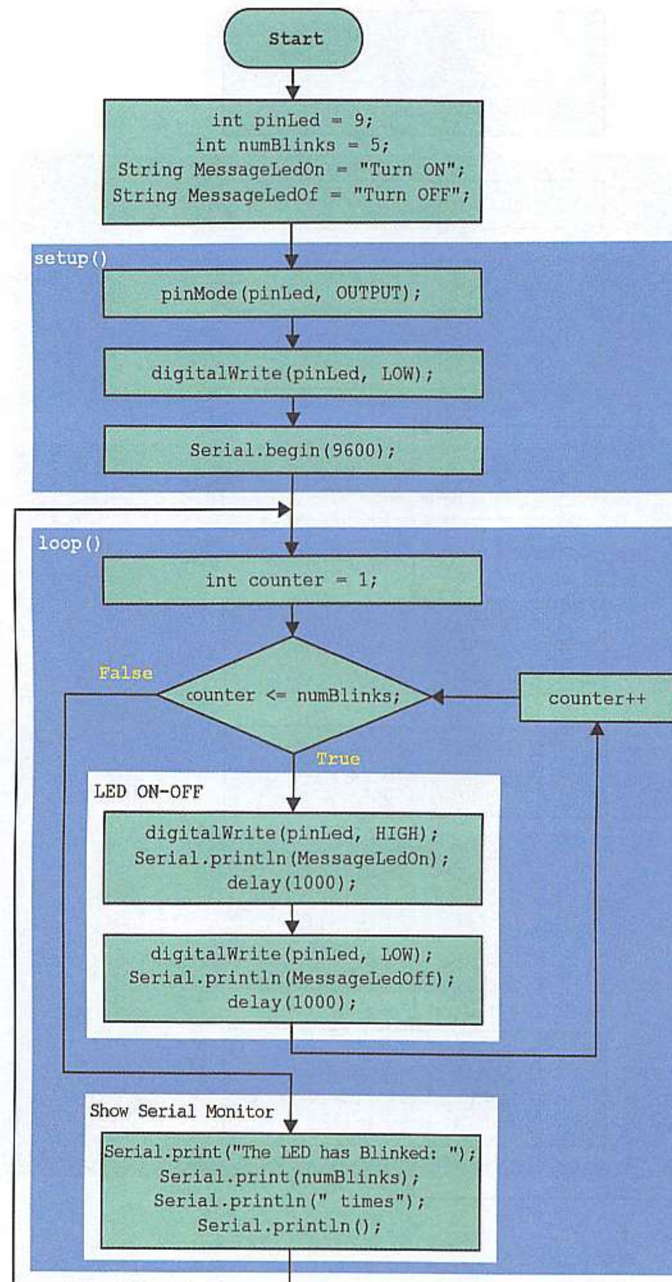
1. บอร์ด Arduino	1	บอร์ด
2. หลอดไฟ LED	1	ดวง
3. ตัวต้านทาน 220 โอห์ม	1	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

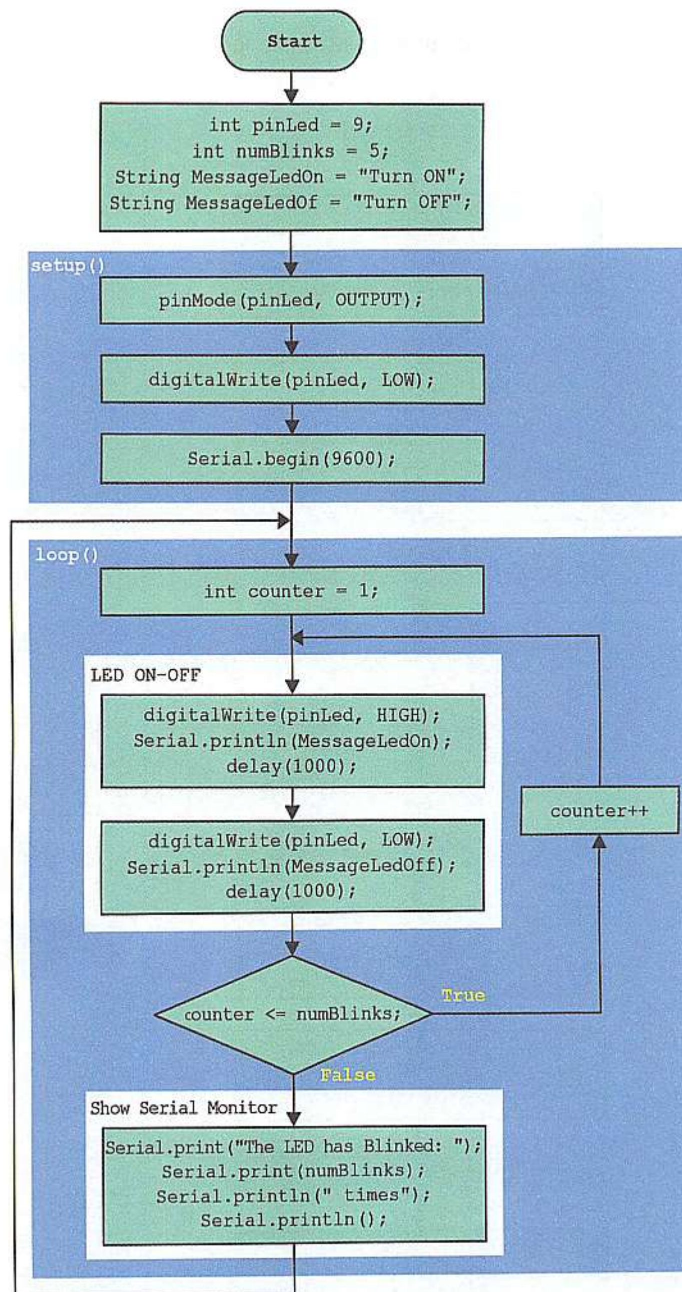
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay() สำหรับ loop, while loop, do while loop

Step 3 : Flowchart Arduino Tutorial 13

แบบ for loop และ while loop



IIUU do while loop



Step 4 : Source Code Arduino_Tutorial_13.ino

13.1 แบบ for loop

```
int pinLed = 9;                 //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
int numBlinks = 5;              //สร้างตัวแปรชื่อ numBlinks ให้เป็น Global Variable เก็บค่า 5
String MessageLedOn = "Turn ON";
    //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ "Turn ON"
String MessageLedOff = "Turn OFF";
    //สร้างตัวแปรชื่อ MessageLedOff ให้เป็น Global Variable เก็บข้อความ "Turn OFF"
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(pinLed, OUTPUT);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);           //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //ฟังก์ชัน loop()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 for (int counter = 1; counter <= numBlinks; counter++) {
    //คำสั่ง for loop ควบคุมรอบไฟกะพริบติดและดับด้วยตัวแปร counter
  digitalWrite(pinLed, HIGH);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println(MessageLedOn);         //แสดงข้อความของตัวแปร MessageLedOn คือ Turn ON
  delay(1000);                         //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println(MessageLedOff);        //แสดงข้อความของตัวแปร MessageLedOn คือ Turn  OFF
  delay(1000);                         //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 }                                     //ออกจากคำสั่ง for loop
 Serial.print("The LED has Blinked: "); //แสดงข้อความ The LED has Blinked:
 Serial.print(numBlinks);       //แสดงค่าที่เก็บในตัวแปร numBlinks เพื่อบอกว่าครบจำนวน 5 รอบแล้ว
 Serial.println(" times");      //แสดงข้อความ times
 Serial.println();              //ขึ้นบรรทัดใหม่ หรือใช้คำสั่ง Serial.print( '\n' ) เพื่อขึ้นบรรทัดใหม่ได้เช่นกัน
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

13.2 แบบ while loop

```
int pinLed = 9;                 //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
int numBlinks = 5;              //สร้างตัวแปรชื่อ numBlinks ให้เป็น Global Variable เก็บค่า 5
String MessageLedOn = "Turn ON";
    //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ Turn ON
String MessageLedOff = "Turn OFF";
    //สร้างตัวแปรชื่อ MessageLedOff ให้เป็น Global Variable เก็บข้อความ Turn OFF
void setup()                    //ฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(pinLed, OUTPUT);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);           //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                     //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
{                               //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 int counter = 1;               //กำหนดตัวแปร counter ค่าเริ่มต้นเท่ากับ 1
 while (counter <= numBlinks) {
    //คำสั่ง while loop เพื่อควบคุมรอบไฟกะพริบวนซ้ำจำนวน 5 รอบ ด้วยเงื่อนไข counter <= numBlinks
  digitalWrite(pinLed, HIGH);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println(MessageLedOn);   //แสดงข้อความของตัวแปร MessageLedOn คือ Turn ON
  delay(1000);                    //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0เพื่อให้หลอดไฟ LED ดับ
  Serial.println(MessageLedOff);  //แสดงข้อความของตัวแปร MessageLedOn คือ Turn OFF
  delay(1000);                    //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 counter++;                       //อัพเดตค่า counter+1 เพื่อวนซ้ำในรอบถัดไป
 }                                //ออกจากคำสั่ง for loop
 Serial.print("The LED has Blinked: ");  //แสดงข้อความ The LED has Blinked:
 Serial.print(numBlinks);
    //แสดงค่าที่เก็บในตัวแปร numBlinks เพื่อบอกว่าครบจำนวน 5 รอบแล้ว
 Serial.println(" times");      //แสดงข้อความ times
 Serial.println();              //ขึ้นบรรทัดใหม่ หรือใช้คำสั่ง Serial.print('\n') เพื่อขึ้นบรรทัดใหม่ได้เช่นกัน
}                               //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

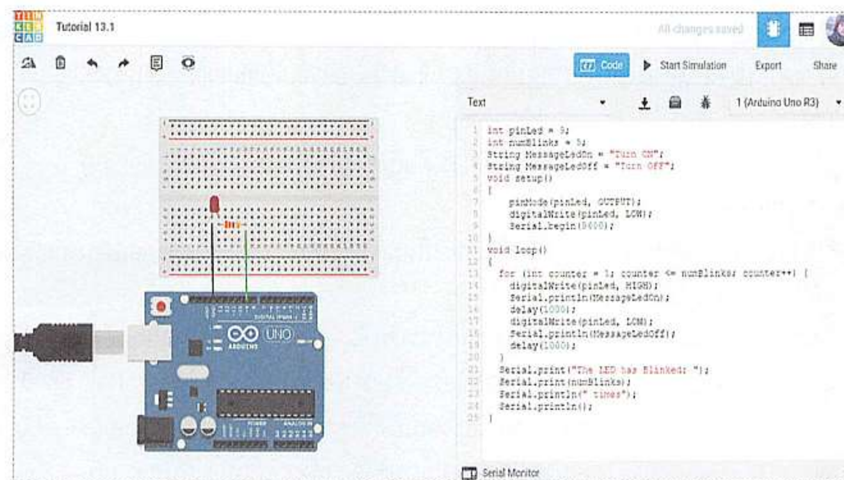
13.3 แบบ do while loop

```
int pinLed = 9;                    //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
int numBlinks = 5;                 //สร้างตัวแปรชื่อ numBlinks ให้เป็น Global Variable เก็บค่า 5
String MessageLedOn = "Turn ON";
    //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ Turn ON
String MessageLedOff = "Turn OFF";
    //สร้างตัวแปรชื่อ MessageLedOff ให้เป็น Global Variable เก็บข้อความ Turn OFF
void setup()                       //ฟังก์ชัน setup()
{                                  //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
  pinMode(pinLed, OUTPUT);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.begin(9600);              //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                                  //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                        //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
{                                  //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 int counter = 1;                  //กำหนดตัวแปร counter ค่าเริ่มต้นเท่ากับ 1
 while (counter <= numBlinks) {
    //คำสั่ง while loop เพื่อควบคุมรอบไฟกะพริบวนซ้ำจำนวน 5 รอบ ด้วยเงื่อนไข counter <= numBlinks
  digitalWrite(pinLed, HIGH);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println(MessageLedOn);    //แสดงข้อความของตัวแปร MessageLedOn คือ Turn ON
  delay(1000);                     //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println(MessageLedOff);   //แสดงข้อความของตัวแปร MessageLedOn คือ Turn OFF
  delay(1000);                     //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 counter++;                        //อัพเดตค่า counter+1 เพื่อวนซ้ำในรอบถัดไป
 }                                 //ออกจากคำสั่ง for loop
 Serial.print("The LED has Blinked: ");  //แสดงข้อความ The LED has Blinked:
 Serial.print(numBlinks);
    //แสดงค่าที่เก็บในตัวแปร numBlinks เพื่อบอกว่าครบจำนวน 5 รอบแล้ว
 Serial.println(" times");         //แสดงข้อความ times
 Serial.println();                 //ขึ้นบรรทัดใหม่ หรือใช้คำสั่ง Serial.print('\n') เพื่อขึ้นบรรทัดใหม่ได้เช่นกัน
}                                  //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

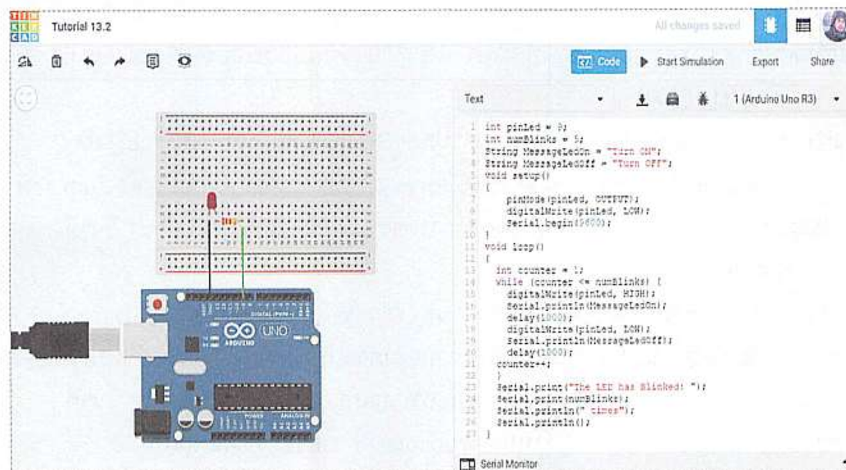
Step 5 : วิธีการทดลอง Arduino Tutorial 13

Tinkercad

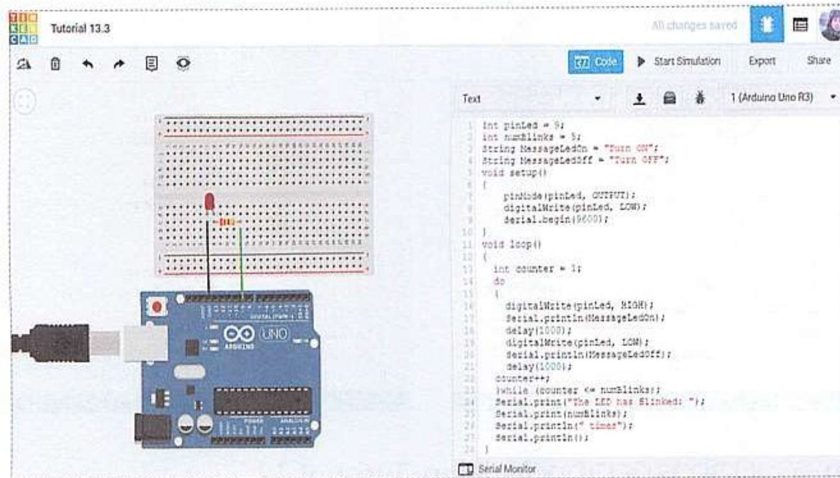
13.1 แบบ for loop



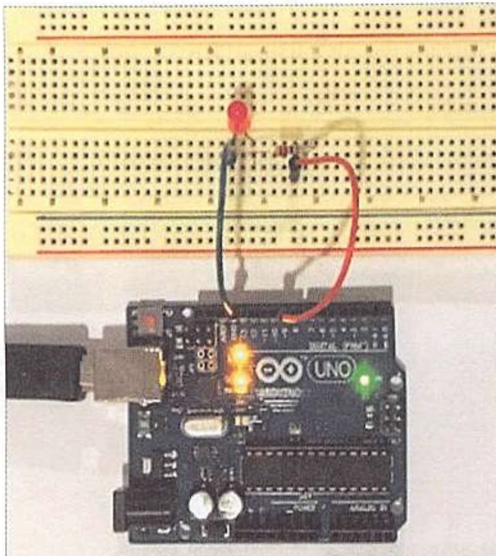
13.2 แบบ while loop



13.3 แบบ do while loop



Arduino Board



for

```
void loop()
{
  for (int counter = 1; counter <= numBlinks; counter++) {
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
  }
  Serial.print("The LED has Blinked: ");
  Serial.print(numBlinks);
  Serial.println(" times");
  Serial.println();
}
```

while

```
}
void loop()
{
  int counter = 1;
  while (counter <= numBlinks) {
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
  counter++;
  }
  Serial.print("The LED has Blinked: ");
  Serial.print(numBlinks);
  Serial.println(" times");
  Serial.println();
}
```

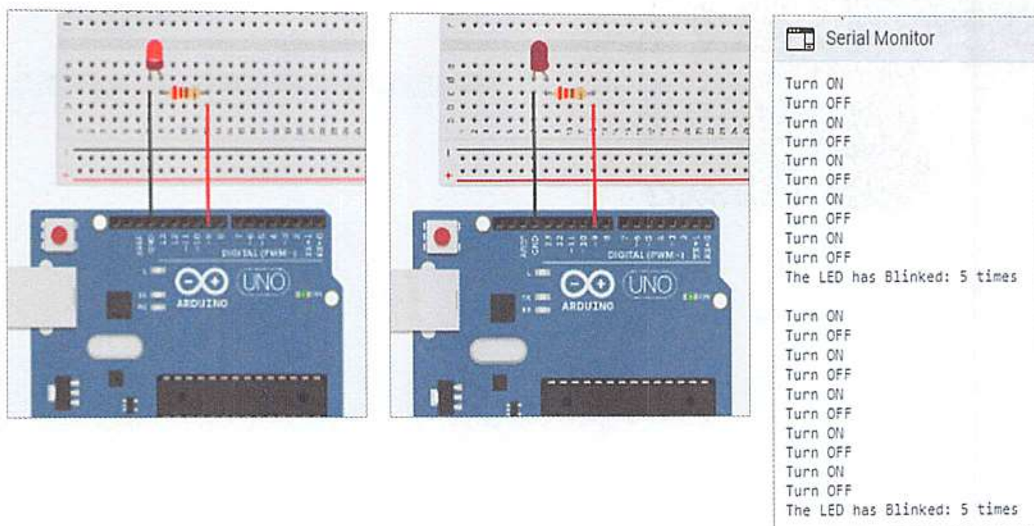
do while

```
void loop()
{
  int counter = 1;
  do
  {
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
  counter++;
  }while (counter <= numBlinks);
  Serial.print("The LED has Blinked: ");
  Serial.print(numBlinks);
  Serial.println(" times");
  Serial.println();
}
```

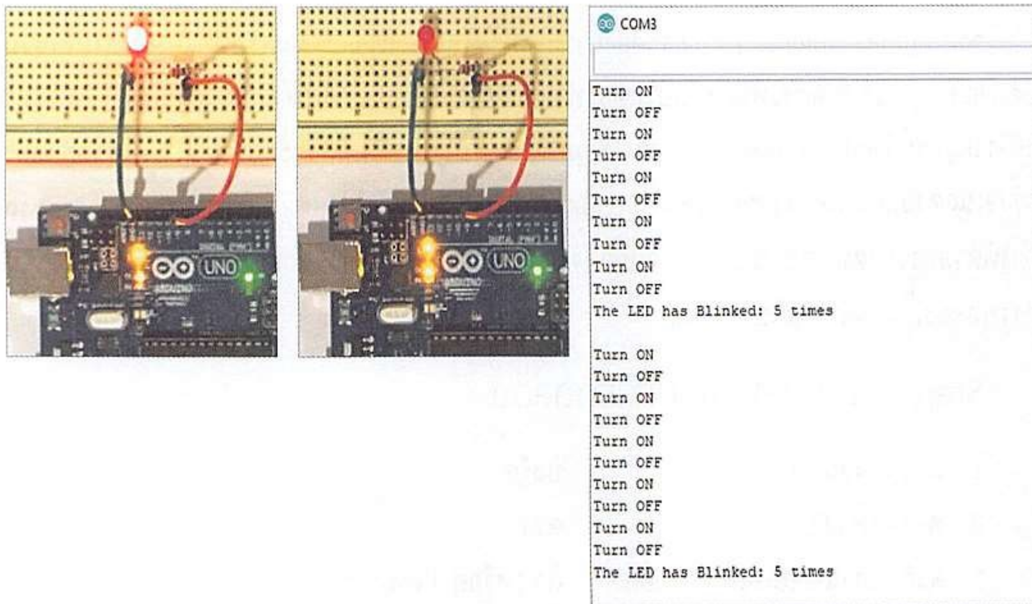
Step 6 : ผลการทดลอง Arduino Tutorial 13

ผลลัพธ์ที่ได้จะแสดงการกะพริบไฟติดและดับจำนวน 5 รอบ แล้วจะแสดงข้อความว่า กะพริบไฟครบ 5 รอบแล้ว และจะทำซ้ำไปเรื่อยๆ เช่นเดิม แต่สิ่งที่แตกต่างกัน คือ เราจะเขียนโปรแกรมได้สั้นกระชับ และยืดหยุ่นมากขึ้น เพราะสามารถนับจำนวนรอบของไฟกะพริบเพียงปรับค่าตัวเลข 5 ให้เป็นจำนวนเต็มตามที่ต้องการ เพื่อแสดงข้อความการนับรอบของไฟกะพริบ โดยไม่จำเป็นต้องไปเพิ่มโค้ดการควบคุมไฟกะพริบเหมือนใน Arduino Tutorial 12 ซึ่งการใช้คำสั่งลูปจะมีประโยชน์มากในการเขียนโปรแกรมที่เริ่มมีความซับซ้อน หรือการสร้างระบบอัตโนมัติ

Tinkercad Output



Arduino Board Output



Step 7 : คำถามท้าย Arduino Tutorial 13

1. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบทุกๆ 3 รอบ โดยใช้คำสั่ง Loop จะเขียนโปรแกรมอย่างไร
2. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบทุกๆ 8 รอบ โดยใช้คำสั่ง Loop จะเขียนโปรแกรมอย่างไร
3. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบครั้งที่ 1 คือ รอบที่ 2, ครั้งที่ 2 คือ รอบที่ 4 และครั้งที่ 3 คือ รอบที่ 6 โดยใช้คำสั่ง Loop จะเขียนโปรแกรมอย่างไร
4. ถ้าต้องการให้แสดงข้อความจำนวนรอบการกะพริบครั้งที่ 1 คือ รอบที่ 3, ครั้งที่ 2 คือ รอบที่ 6 และครั้งที่ 3 คือ รอบที่ 9 โดยใช้คำสั่ง Loop จะเขียนโปรแกรมอย่างไร

Arduino Tutorial 14 การปรับแต่งโปรแกรมไฟกะพริบ

จาก Arduino Tutorial 13 ถ้าเราต้องการที่จะทำให้หลอดไฟ LED กะพริบติดและดับตามจำนวนที่ต้องการ แล้วให้หยุดการทำงานโดยไม่ต้องทำงานซ้ำต่อจะทำอย่างไร ใน Arduino Tutorial 14 จะมาแก้ไขปัญหานี้ โดยในการทดลองนี้เราจะควบคุมไฟกะพริบให้กะพริบติดและดับจำนวน 5 ครั้ง โดยแต่ละครั้งจะแสดงข้อความจำนวนรอบแต่ละรอบ และเมื่อครบ 5 รอบแล้วให้หยุดการทำงานทันที เราสามารถทำได้ด้วยการกำหนดจำนวนรอบการวนซ้ำ โดยการใช้ตัวแปร counter มาช่วยควบคุมเงื่อนไข โดยในการทดลองนี้จะใช้คำสั่งแบบ for loop

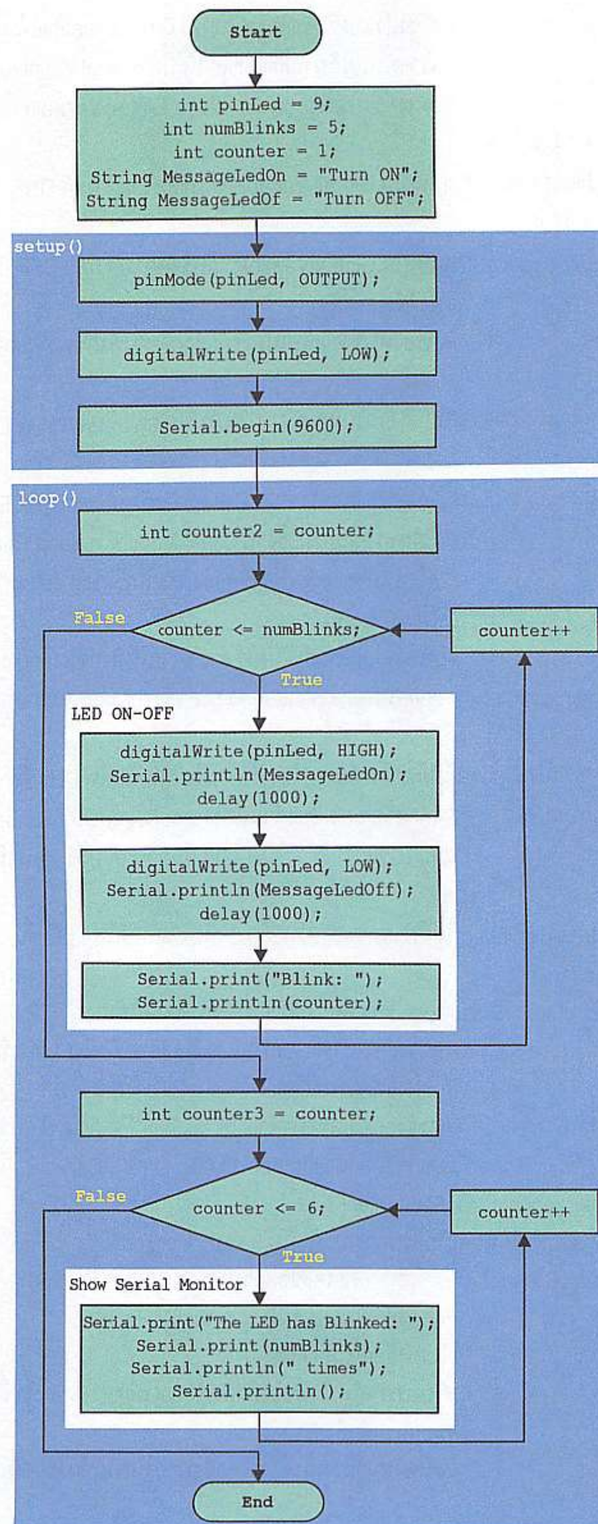
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino 1 บอร์ด
2. หลอดไฟ LED 1 ดวง
3. ตัวต้านทาน 220 โอห์ม 1 ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay(), for loop
```

Step 3 : Flowchart Arduino Tutorial 14

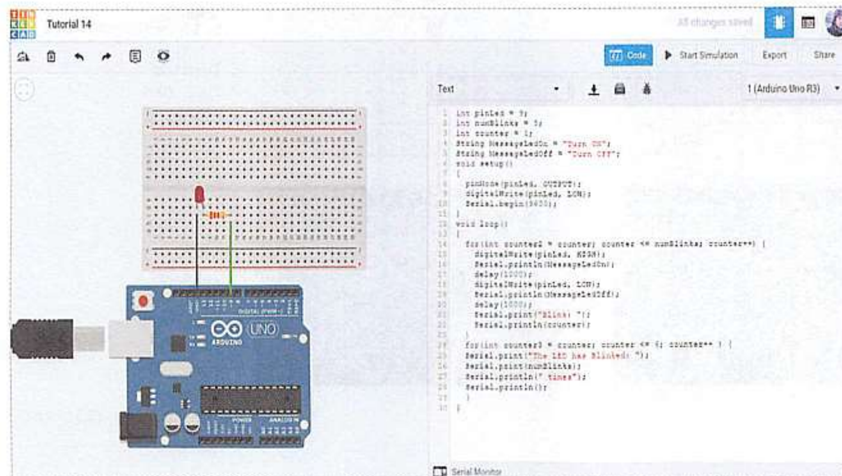


Step 4 : Source Code Arduino_Tutorial_14.ino

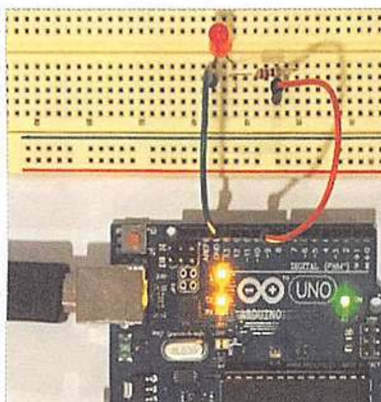
```
int pinLed = 9;                    //สร้างตัวแปรชื่อ pinLed ให้เป็น Global Variable ให้ใช้ Pin 9
int numBlinks = 5;                 //สร้างตัวแปรชื่อ numBlinks ให้เป็น Global Variable เก็บค่า 5
int counter = 1;                   //สร้างตัวแปรชื่อ counter ให้เป็น Global Variable เก็บค่า 1
String MessageLedOn = "Turn ON";
        //สร้างตัวแปรชื่อ MessageLedOn ให้เป็น Global Variable เก็บข้อความ "Turn ON"
String MessageLedOff = "Turn OFF";
        //สร้างตัวแปรชื่อ MessageLedOff ให้เป็น Global Variable เก็บข้อความ "Turn OFF"
void setup()                       //ฟังก์ชัน setup()
{                                  //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 pinMode(pinLed, OUTPUT);
        //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้ทำงานในโหมด OUTPUTเพื่อแสดงค่าออกทางหลอดไฟ LED
 digitalWrite(pinLed, LOW);
        //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะเริ่มต้น LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
 Serial.begin(9600);               //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                                  //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop()                        //ฟังก์ชัน loop()
{                                  //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน loop()
 for(int counter2 = counter; counter <= numBlinks; counter++) { //คำสั่ง for loop 1 ควบคุมไฟกะพริบติดและดับ
  digitalWrite(pinLed, HIGH);
        //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  Serial.println(MessageLedOn);    //แสดงข้อความของตัวแปร MessageLedOn คือ Turn ON ขึ้นบรรทัดใหม่
  delay(1000);                     //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed, LOW);
        //ตั้งขา Pin 9 ผ่านตัวแปร pinLed ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  Serial.println(MessageLedOff);
        //แสดงข้อความของตัวแปร MessageLedOn คือ Turn  OFF ขึ้นบรรทัดใหม่
  delay(1000);                     //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  Serial.print("Blink: ");         //แสดงข้อความ Blink:
  Serial.println(counter);         //แสดงค่าตัวเลขของตัวแปร  counter คือ ตัวเลขรอบของการกะพริบไฟ
 }                                 //ออกจากคำสั่ง for loop 1
 for(int counter3 = counter; counter <= 6; counter++ ) {
        //คำสั่ง for loop 2 ควบคุมให้หยุดเมื่อครบการกะพริบไฟ 5 รอบ
 Serial.print("The LED has Blinked:  ");       //แสดงข้อความ The LED has Blinked:
 Serial.print(numBlinks);          //แสดงค่าตัวเลขของตัวแปร numBlinks คือ 5
 Serial.println("  times");        //แสดงข้อความ times
 Serial.println();                 //ขึ้นบรรทัดใหม่ หรือใช้คำสั่ง Serial.print( '\n' ) เพื่อขึ้นบรรทัดใหม่ได้เช่นกัน
 }                                 //ออกจากคำสั่ง for loop 2
}                                  //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 14

Tinkercad



Arduino Board

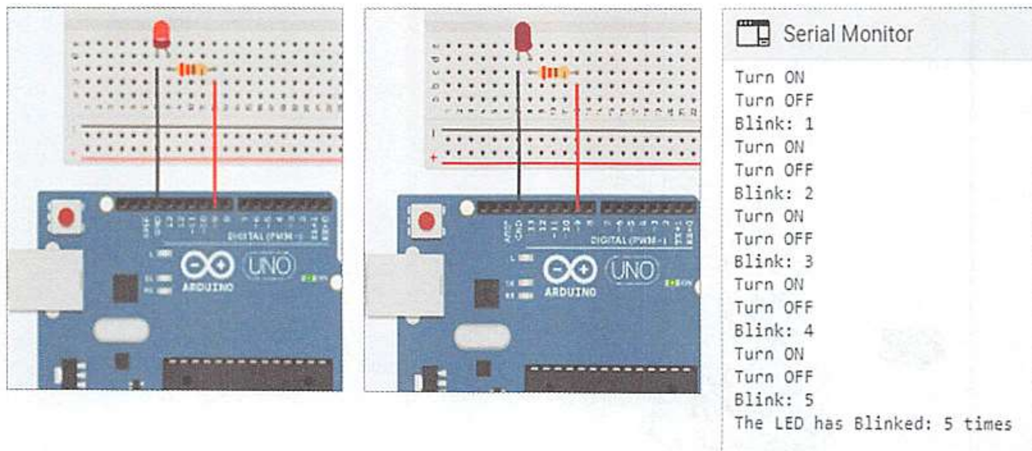


Arduino_Tutorial_14

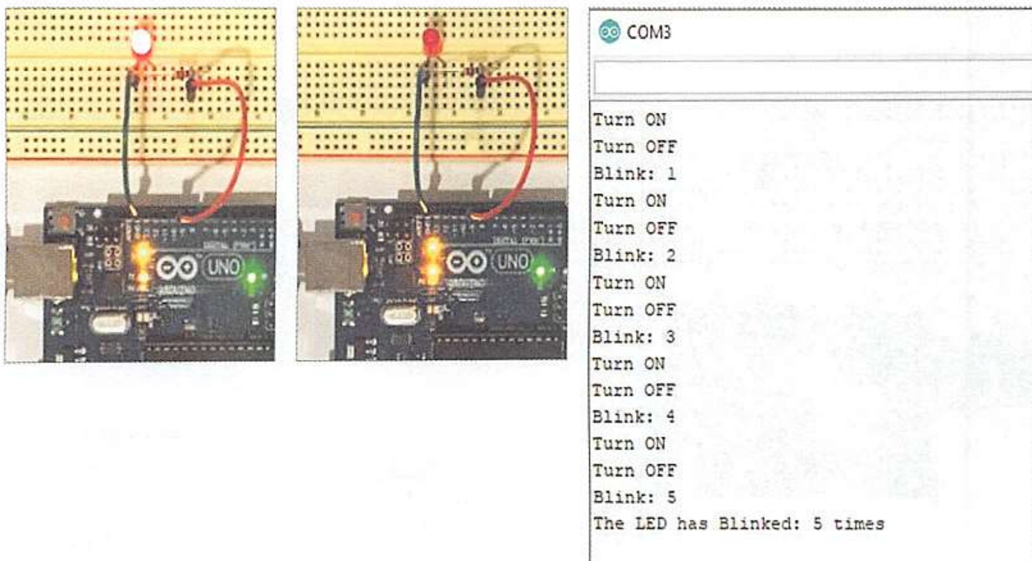
```
int pinLed = 9;
int numBlinks = 5;
int counter = 1;
String MessageLedOn = "Turn ON";
String MessageLedOff = "Turn OFF";
void setup()
{
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  Serial.begin(9600);
}
void loop()
{
  for(int counter2 = counter; counter <= numBlinks; counter++) {
    digitalWrite(pinLed, HIGH);
    Serial.println(MessageLedOn);
    delay(1000);
    digitalWrite(pinLed, LOW);
    Serial.println(MessageLedOff);
    delay(1000);
    Serial.print("Blink: ");
    Serial.println(counter);
  }
  for(int counter3 = counter; counter <= 6; counter++ ) {
  Serial.print("The LED has Blinked: ");
  Serial.print(numBlinks);
  Serial.println(" times");
```

Step 6 : ผลการทดลอง Arduino Tutorial 14

Tinkercad Output



Arduino Board Output



ผลลัพธ์ที่ได้จะแสดงการกะพริบไฟติดและดับจำนวน 5 ครั้ง โดยแต่ละรอบจะแสดงข้อความการติดและดับพร้อมจำนวนรอบ เมื่อวนรอบจนครบ 5 ครั้งแล้วจะแสดงข้อความ The LED has Blinked: 5 times และจบการทำงานโดยไม่มีการวนซ้ำใหม่

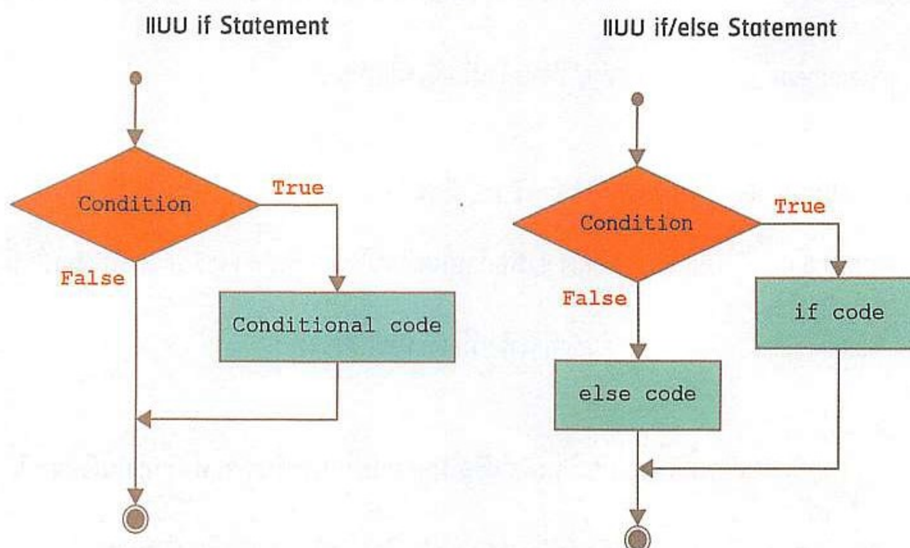
Step 7 : คำถามท้าย Arduino Tutorial 14

1. ทดลองเปลี่ยนใช้เฉพาะคำสั่ง while loop ผลลัพธ์จะเหมือนเดิมหรือไม่
2. ทดลองเปลี่ยนใช้เฉพาะคำสั่ง do while loop ผลลัพธ์จะเหมือนเดิมหรือไม่
3. ทดลองเปลี่ยนใช้คำสั่งผสมระหว่าง for loop และ while loop ผลลัพธ์จะเหมือนเดิมหรือไม่
4. ถ้าต้องการให้โปรแกรมแสดงไฟกะพริบติดและดับจำนวน 9 รอบ แล้วหยุดการทำงานทันที จะต้องเขียนโปรแกรมอย่างไร

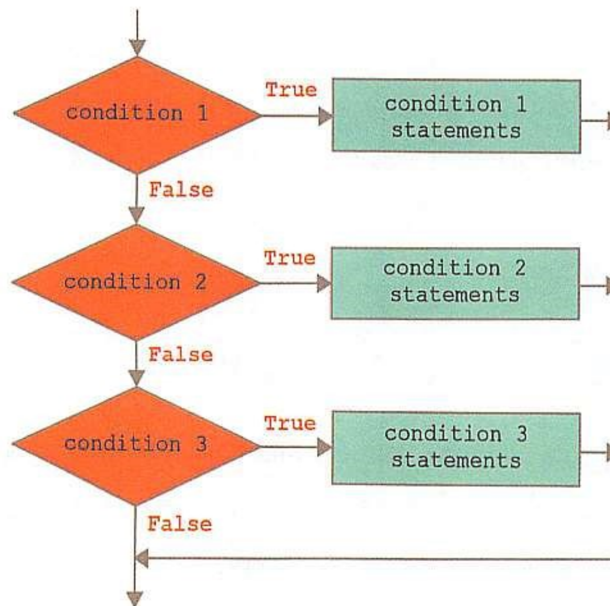
6.9 การใช้คำสั่งเงื่อนไข if/else

ในการเขียนโปรแกรมควบคุมการทำงานบอร์ด Arduino โปรแกรมเมอร์จะต้องมีทักษะในการเขียนโปรแกรมที่สามารถระบุเงื่อนไขหนึ่งหรือมากกว่านั้น เช่น เมื่อผู้ใช้พิมพ์คำว่า red สั่งให้หลอดไฟ LED สีแดงกะพริบ หรือพิมพ์คำว่า green สั่งให้หลอดไฟ LED สีเขียวกะพริบ ทั้งนี้เพื่อเป็นออปชันให้ผู้ใช้เลือกเงื่อนไข หรือเพื่อใช้ประโยชน์ในการทดสอบโปรแกรมสำหรับผู้พัฒนาโปรแกรม เราจะใช้คำสั่งเงื่อนไข if/else มาช่วยในการตัดสินใจดังกล่าว

คำสั่ง if/else เป็นคำสั่งที่ใช้ตัดสินใจเลือกเงื่อนไขที่ต้องการว่า ถ้าเงื่อนไขเป็นจริงจะทำคำสั่งนี้ ถ้าเงื่อนไขเป็นเท็จจะทำอีกเงื่อนไขหนึ่ง



แบบ if... else if... else Statement



Syntax : แบบ if Statement ใช้สำหรับ 1 เงื่อนไข

(สามารถใช้มากกว่า 1 เงื่อนไขได้เช่นเดียวกับ if/else จะแสดงใน Arduino Tutorial 15)

```
if (condition)          //คำสั่ง if เพื่อกำหนดเงื่อนไข ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้
{
   Statements;          //คำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง
}
```

Syntax : แบบ if/else Statement ใช้สำหรับ 2 เงื่อนไข

```
if (condition)          //คำสั่ง if เพื่อกำหนดเงื่อนไข ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้
{
   Statements;          //คำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง
}
else
      //คำสั่ง else กรณีที่เงื่อนไขเป็นเท็จ หรือไม่ตรงกับเงื่อนไขอื่นๆ จะทำคำสั่งในเงื่อนไขนี้
{
   Statements;          //คำสั่งที่จะทำเมื่อเป็นเท็จ หรือไม่ตรงกับเงื่อนไขอื่นๆ
}
```

Syntax : แบบ if...else if...else Statement ใช้ตั้งแต่ 3 เงื่อนไขขึ้นไป

```
if (condition)        //คำสั่ง if เพื่อกำหนดเงื่อนไข ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้
{
    Statements;       //คำสั่งที่จะทำเมื่อเงื่อนไข if เป็นจริง
}
else if (condition)   //คำสั่ง else if เพื่อกำหนดเงื่อนไข ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้
{
    Statements;       //คำสั่งที่จะทำเมื่อเงื่อนไข else if เป็นจริง
}
else                  //คำสั่ง else กรณีที่ไม่ตรงกับเงื่อนไขอื่นๆ จะทำคำสั่งในเงื่อนไขนี้
{
    Statements;       //คำสั่งที่จะทำเมื่อไม่ตรงกับเงื่อนไขอื่นๆ
}
```

Parameters :

condition : กำหนดเงื่อนไข ถ้าเป็นจริงจะทำตามคำสั่งในเงื่อนไขนี้

Arduino Tutorial 15 ทดลองใช้คำสั่ง if/else เพื่อสั่งหลอดไฟ LED ที่ต้องการให้กะพริบ

ใน Arduino Tutorial ที่ 14 เราได้ใช้ Input ที่มาจากการใช้คำสั่ง loop เพื่อให้แสดงการกะพริบไฟผ่าน LED และข้อความผ่าน Serial Monitor ใน Arduino Tutorial นี้เราจะให้ผู้ใช้ป้อนข้อมูลสีหลอดไฟ LED ที่ต้องการ (Input) ด้วยตนเอง โดยเราจะใช้ LED 2 ดวง คือ สีแดงและสีเขียว เป็นตัวอย่างเริ่มต้นสำหรับการเรียนรู้การใช้คำสั่งเงื่อนไข if/else โดยมีเงื่อนไขถ้าผู้ใช้พิมพ์ "red" จะเลือกควบคุมให้หลอดไฟ LED สีแดงกะพริบ หรือถ้าผู้ใช้พิมพ์ "green" จะเลือกควบคุมให้หลอดไฟ LED สีเขียวกะพริบ แต่ถ้าเลือกสีอื่นหรือพิมพ์ไม่ถูกต้อง หลอดไฟ LED ทั้งสองสีจะไม่ติด และแสดงข้อความแจ้งเตือนให้ผู้ใช้ใส่ข้อมูลสีให้ถูกต้อง ในการทดลองนี้จะแสดงการเขียนโปรแกรมทั้งแบบ if Statement และ if/else Statement ซึ่งผลลัพธ์จะได้เช่นเดียวกัน ดังนี้

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino 1 บอร์ด
2. หลอดไฟ LED 2 ดวง
3. ตัวต้านทาน 220 โอห์ม 2 ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

setup(): pinMode(), digitalWrite(), Serial.begin()

loop(): digitalWrite(), Serial.println(), delay(), Serial.available(), Serial.readStringUntil()

If Statement: if(), if()...else if()...else

รายละเอียดการใช้งานคำสั่ง

Serial.available() คือ เป็นคำสั่งที่ใช้ตรวจสอบว่า มีข้อมูลเข้ามาแล้วหรือยัง เช่น while (Serial.available() == 0) { } คือ ตรวจสอบว่า ถ้ายังไม่มีการรับข้อมูลสถานะจะรอไปเรื่อยๆ หรือ If(Serial.available > 0) คือ ถ้าตรวจสอบแล้วพบว่า มีข้อมูลมาแล้วจะให้ทำตามคำสั่งที่กำหนดต่อไป

Serial.readStringUntil() คือ คำสั่งที่ใช้อ่านค่าข้อมูลชนิด String จนถึงค่าตัวอักขระสุดท้าย เช่น '\n' เช่น ใน Serial Monitor ของโปรแกรม Arduino IDE จะตั้งค่า Default ว่า Newline คือ เมื่อส่งข้อความมาแล้วจะขึ้นบรรทัดใหม่ให้ คือ ตัวอักขระ '\n' ถึงจะอ่านค่าข้อมูล

Syntax : Serial.readStringUntil(terminator)

Parameters : serial คือ พอร์ตอนุกรม ดูรายการพอร์ตอนุกรมของบอร์ดแต่ละรุ่นได้จากเว็บไซต์ https://www.arduino.cc/reference/en/language/functions/communication/serial/

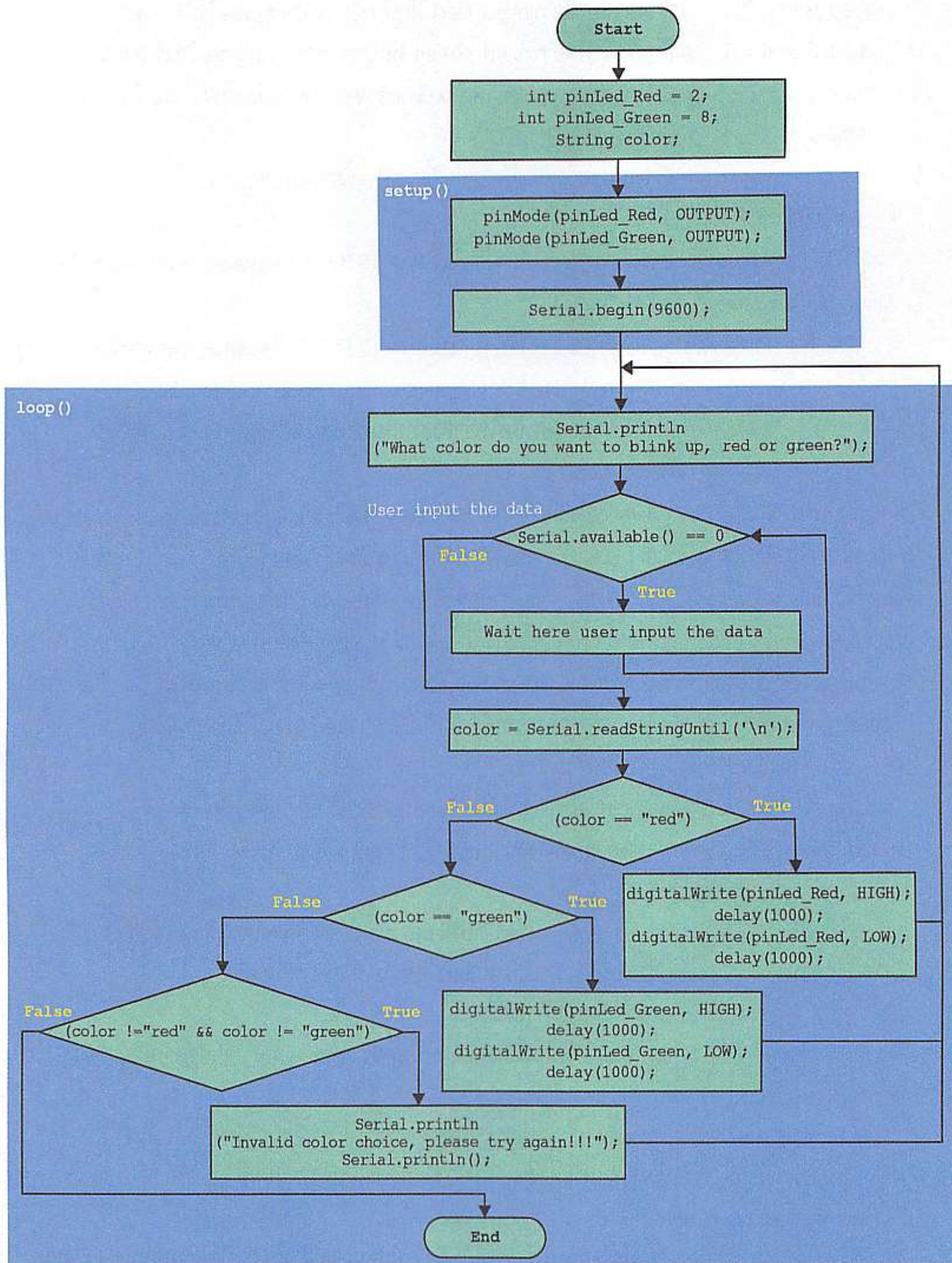
terminator : ชนิดข้อมูลที่ใช้ได้ คือ ชนิดข้อมูลแบบตัวอักษร; Data Types: char

Serial.readString() คำสั่งที่ใช้อ่านค่าข้อมูลชนิด String อย่างเดียว โดยไม่มีตัวอักขระตามหลัง เช่น ใน Serial Monitor ของโปรแกรม Arduino IDE จะต้องตั้งในโหมด No line ending ถึงจะอ่านค่าข้อมูลได้

Syntax : Serial.readString()

Parameters : serial: พอร์ตอนุกรม ดูรายการพอร์ตอนุกรมของบอร์ดแต่ละรุ่นได้จากเว็บไซต์ https://www.arduino.cc/reference/en/language/functions/communication/serial/

Step 3 : Flowchart Arduino Tutorial 15 แบบ if Statement



Step 4 : Source Code Arduino_Tutorial_15.ino

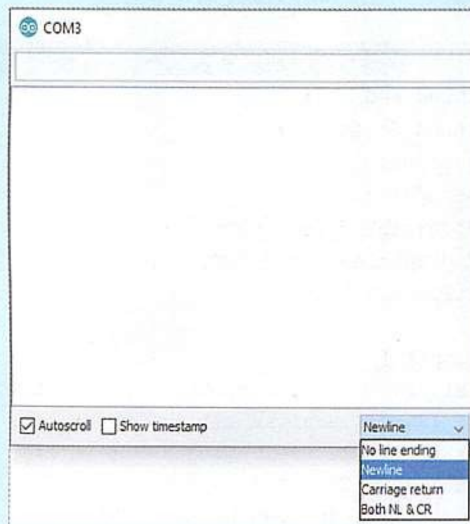
```
int pinLed_Red = 2;      //สร้างตัวแปรชื่อ pinLed_Red ให้เป็น Global Variable ให้ใช้ Pin 2
int pinLed_Green = 8;    //สร้างตัวแปรชื่อ pinLed_Green ให้เป็น Global Variable ให้ใช้ Pin 8
String color;            //สร้างตัวแปรชื่อ color ให้เป็น Global Variable ชนิด String เก็บข้อความ
void setup()             //ฟังก์ชัน setup()
{                        //เครื่องหมายวงเล็บเปิดใช้คำสั่งภายในฟังก์ชัน setup()
 pinMode(pinLed_Red, OUTPUT);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed_Red ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
pinMode(pinLed_Green, OUTPUT);
    //ตั้งขา Pin 9 ผ่านตัวแปร pinLed_Green ให้ทำงานในโหมด OUTPUT เพื่อแสดงค่าออกทางหลอดไฟ LED
 Serial.begin(9600);     //เปิดพอร์ตอนุกรมใช้งาน Serial Monitor ที่ Baud Rate 9600
}                        //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน setup()
void loop() {            //ฟังก์ชัน loop()
 Serial.println("What color do you want to blink up, red or green?"); //แสดงข้อความและขึ้นบรรทัดใหม่
 while (Serial.available() == 0) { }        //คำสั่งสำหรับรอการป้อนข้อมูล Input
 color = Serial.readStringUntil( '\n' );    //รับค่าข้อความจากคีย์บอร์ดมาเก็บไว้ที่ตัวแปร color
    //คำสั่ง Serial.readStringUntil( '\n' ) ใช้ในกรณีที่ Serial Monitor ตั้งค่าเป็น Newline
    //หรือถ้า Serial Monitor ตั้งค่าเป็น No line ending สามารถใช้คำสั่ง Serial.readString() ได้เช่นเดียวกัน
 if (color == "red") {   //คำสั่ง if ตรวจสอบเงื่อนไข ถ้าพิมพ์คำว่า red จะทำคำสั่งตามเงื่อนไขนี้
  digitalWrite(pinLed_Red, HIGH);
    //ตั้งค่าขา Pin 2 ผ่านตัวแปร pinLed_Red ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  delay(1000);           //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed_Red, LOW);
    //ตั้งค่าให้ขา Pin 2 ผ่านตัวแปร pinLed_Red ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  delay(1000);           //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 }                       //ออกจากเงื่อนไขคำสั่ง if
 if (color == "green") { //คำสั่ง if ตรวจสอบเงื่อนไข ถ้าพิมพ์คำว่า green จะทำคำสั่งตามเงื่อนไขนี้
  digitalWrite(pinLed_Green, HIGH);
    //ตั้งค่าขา Pin 8 ผ่านตัวแปร pinLed_Green ให้มีค่าสถานะ HIGH คือ 1 เพื่อให้หลอดไฟ LED ติด
  delay(1000);           //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
  digitalWrite(pinLed_Green, LOW);
    //ตั้งค่าให้ขา Pin 2 ผ่านตัวแปร pinLed_Green ให้มีค่าสถานะ LOW คือ 0 เพื่อให้หลอดไฟ LED ดับ
  delay(1000);           //หน่วงเวลาการทำงาน 1,000 มิลลิวินาที หรือ 1 วินาที
 }                       //ออกจากเงื่อนไขคำสั่ง if
```

```
  if (color != "red" && color != "green") {
      //คำสั่ง if ตรวจสอบเงื่อนไข ถ้าพิมพ์ข้อความแต่ไม่ใช่ทั้ง red และ green จะทำเงื่อนไขนี้
    Serial.println("Invalid color choice, please try again!!!"); //แสดงข้อความและขึ้นบรรทัดใหม่
    Serial.println();            //ขึ้นบรรทัดใหม่
  }                              //ออกจากเงื่อนไขคำสั่ง if
}                                //เครื่องหมายวงเล็บปิดใช้คำสั่งภายในฟังก์ชัน loop()
```



Tips & Tricks ในการป้อนข้อมูล Input ของ Serial Monitor ในโปรแกรม Arduino IDE

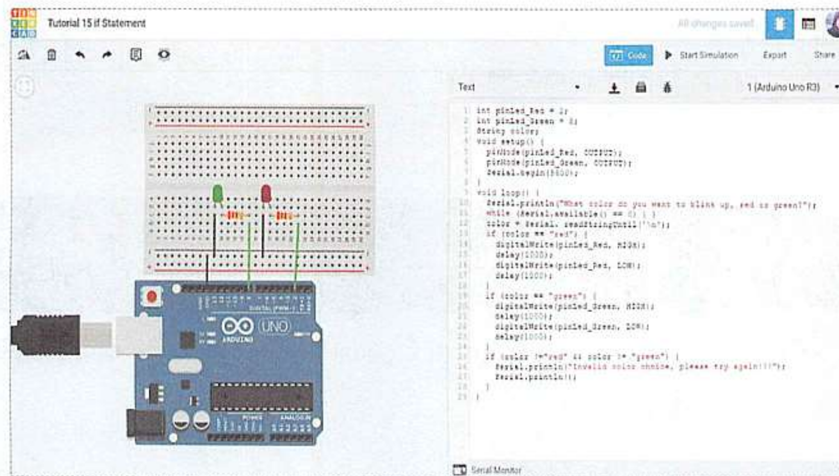
- คำสั่ง Serial.readStringUntil('\n') ใช้ในกรณีที่ Serial Monitor ตั้งค่าเป็น Newline เนื่องจากคำสั่งจะอ่านจนถึงเครื่องหมาย \n
- คำสั่ง Serial.readString() ใช้ในกรณีที่ Serial Monitor ตั้งค่าเป็น No line ending ทั้ง 2 คำสั่งสามารถใช้งานได้เหมือนกัน



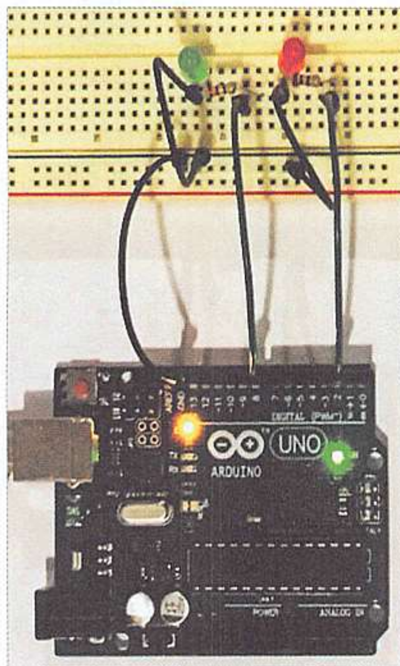
- ใน Serial Monitor จะมีตัวเลือกโหมดในการส่ง โดยเป็นตัวเลือกว่า จะให้มีอะไรพ่วงท้ายข้อมูลไปด้วยหรือไม่ จะมี 4 โหมด
- No line ending ส่งเฉพาะข้อมูล ไม่มีอะไรพ่วงไปด้วย
- Newline ส่งข้อมูลตามด้วย '\n' เช่น พิมพ์คำว่า hello แล้วคลิกปุ่ม Send คอมพิวเตอร์จะส่ง 'h' 'e' 'l' 'l' 'o' '\n' ไปที่ Arduino (เป็นค่า Default ของ Serial Monitor)
- Carriage return ส่ง '\r' พ่วงท้ายข้อมูล
- Both NL & CR ส่ง '\r'และ '\n' พ่วงท้ายข้อมูล

Step 5 : วิธีการทดลอง Arduino Tutorial 15 if Statement

Tinkercad



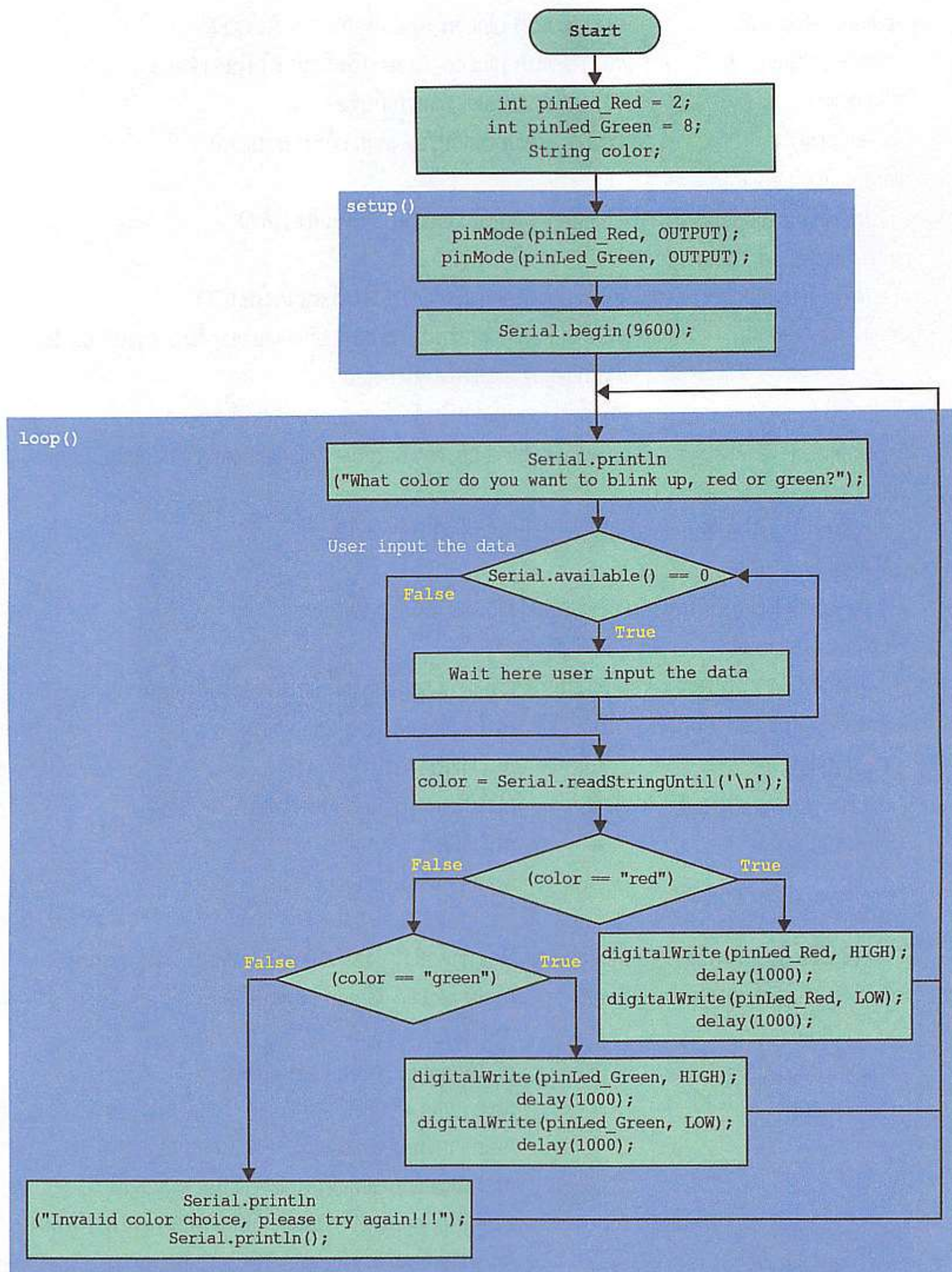
Arduino Board



Arduino_Tutorial_15_1_if

```
int pinLed_Red = 2;
int pinLed_Green = 8;
String color;
void setup() {
  pinMode(pinLed_Red, OUTPUT);
  pinMode(pinLed_Green, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  Serial.println("What color do you want to
  while (Serial.available() == 0) { }
  color = Serial. readStringUntil('\n');
  if (color == "red") {
    digitalWrite(pinLed_Red, HIGH);
    delay(1000);
    digitalWrite(pinLed_Red, LOW);
    delay(1000);
  }
```

Step 6 : Flowchart Arduino Tutorial 15 แบบ if/else Statement

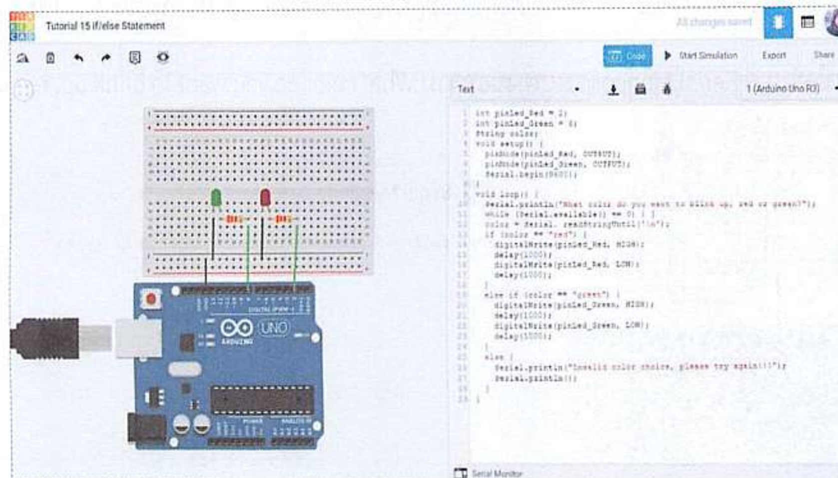


Step 7 : Source Code Arduino Tutorial 15 แบบ if/else Statement

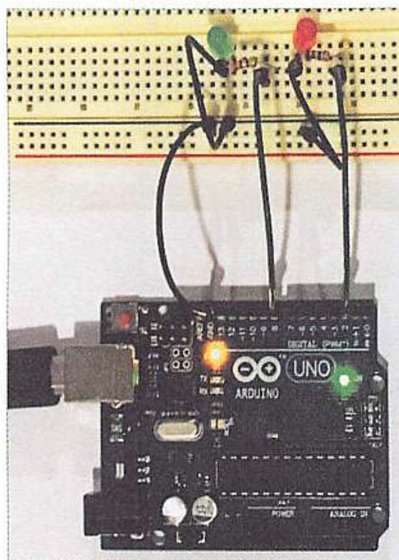
```
int pinLed_Red = 2;            //สร้างตัวแปร pinLed_Red เพื่อกำหนดให้ใช้ขา Pin 2
int pinLed_Green = 8;          //สร้างตัวแปร pinLed_Green เพื่อกำหนดให้ใช้ขา Pin 8
String color;                  //สร้างตัวแปร color สำหรับเก็บข้อความ
void setup() {                 //ฟังก์ชัน setup() สำหรับตั้งค่าการใช้งานอุปกรณ์
 pinMode(pinLed_Red, OUTPUT);
    //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Red เป็น OUTPUT เพื่อแสดงไฟ LED
 pinMode(pinLed_Green, OUTPUT);
    //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Green เป็น OUTPUT เพื่อแสดงไฟ LED
 Serial.begin(9600);   //ฟังก์ชัน Serial.begin สำหรับเปิดใช้งาน Serial Monitor ด้วย Baud Rate 9600
}                              //จบการทำงานของฟังก์ชัน setup()
void loop() {                  //ฟังก์ชัน loop() สำหรับสั่งให้ Arduino ทำงานที่ต้องการ
 Serial.println("What color do you want to blink up, red or green?"); //แสดงข้อความและขึ้นบรรทัดใหม่
 while (Serial.available() == 0) { }
    //เช็คว่ามีการป้อนข้อมูลมาหรือไม่ ถ้ายังไม่มีจะค้างไว้รอจนกว่าจะมี
 color = Serial.readStringUntil('\n');
    //อ่านค่าข้อความที่พิมพ์เข้ามาจาก Serial Monitor โหมด Newline
 if (color == "red") {
    //เช็คเงื่อนไขว่าพิมพ์คำว่า red ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้ ถ้าไม่ใช่จะไปยังเงื่อนไขถัดไป
  digitalWrite(pinLed_Red, HIGH);      //สั่งให้ pinLed_Red แสดงไฟติด
  delay(1000);                         //หน่วงเวลา 1 วินาที
  digitalWrite(pinLed_Red, LOW);       //สั่งให้ pinLed_Red แสดงไฟดับ
  delay(1000);                         //หน่วงเวลา 1 วินาที
 }                                     //ออกจากเงื่อนไข if
 else if (color == "green") {
    //เช็คเงื่อนไขว่าพิมพ์คำว่า green ถ้าเป็นจริงจะทำคำสั่งในเงื่อนไขนี้ ถ้าไม่ใช่จะไปยังเงื่อนไขถัดไป
  digitalWrite(pinLed_Green, HIGH);    //สั่งให้ pinLed_Green แสดงไฟติด
  delay(1000);                         //หน่วงเวลา 1 วินาที
  digitalWrite(pinLed_Green, LOW);     //สั่งให้ pinLed_Green แสดงไฟดับ
  delay(1000);                         //หน่วงเวลา 1 วินาที
 }                                     //ออกจากเงื่อนไข else if
 else {                                //นอกเหนือเงื่อนไขทั้งหมด จะทำงานตามคำสั่งนี้
      Serial.println("Invalid color choice, please try again!!!"); //แสดงข้อความและขึ้นบรรทัดใหม่
      Serial.println();                //ขึ้นบรรทัดใหม่
     }                                 //ออกจากเงื่อนไข else
    }
```

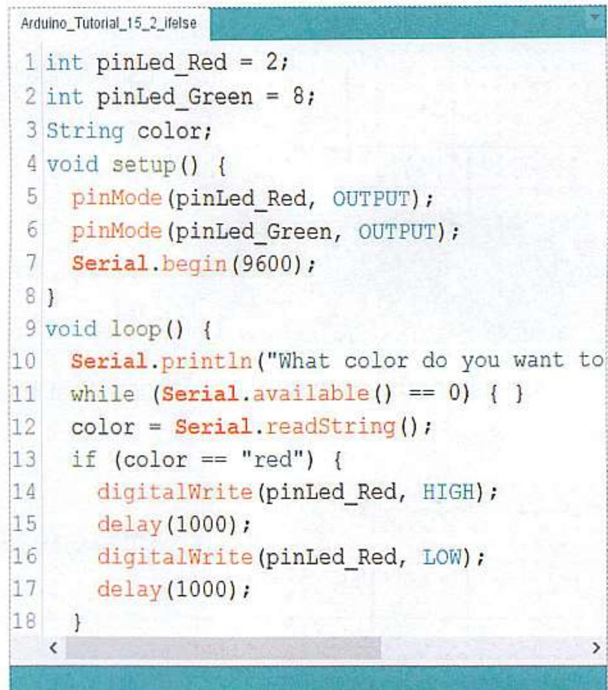
Step 8 : วิธีการทดลอง Arduino Tutorial 15 if/else Statement

Tinkercad



Arduino Board



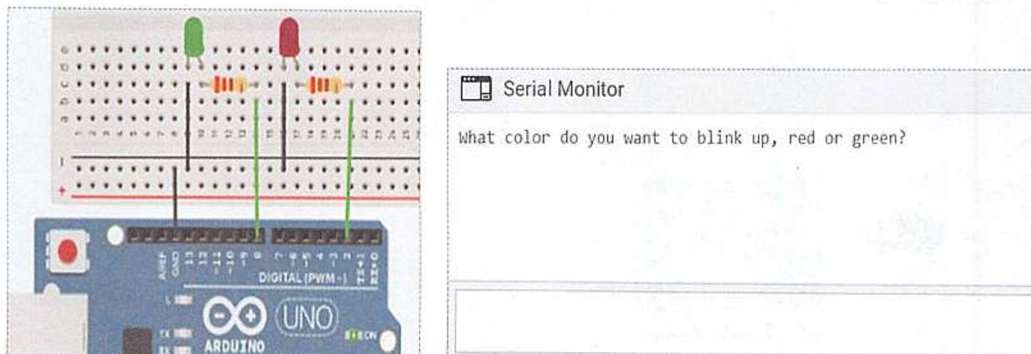


Step 9 : ผลการทดลอง Arduino Tutorial 15

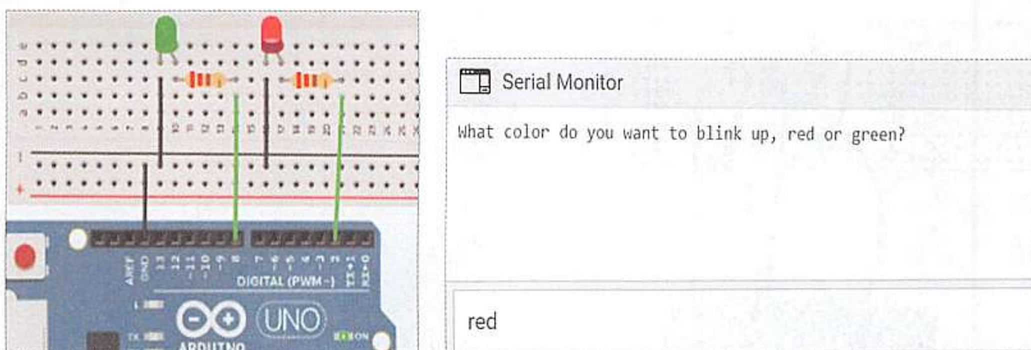
Tinkercad Output

Arduino Tutorial 15 แบบ if Statement และ if/else Statement ผลลัพธ์จะเหมือนกัน

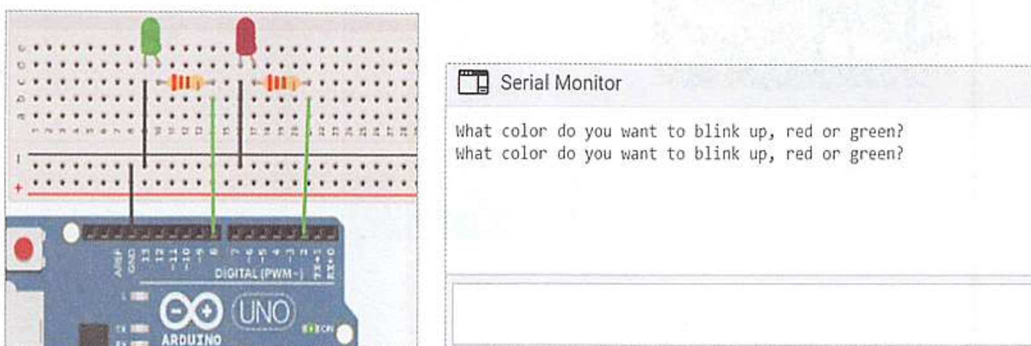
1. เมื่อรันผล Serial Monitor จะแสดงข้อความ What color do you want to blink up, red or green?



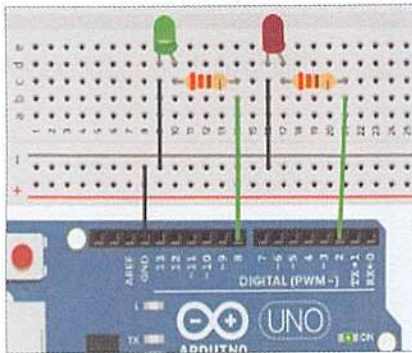
2. ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำว่า red แล้วคลิกปุ่ม Send หลอดไฟ LED สีแดงจะกะพริบ



3. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง



4. ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำว่า green แล้วคลิกปุ่ม Send หลอดไฟ LED สีเขียวจะกะพริบ

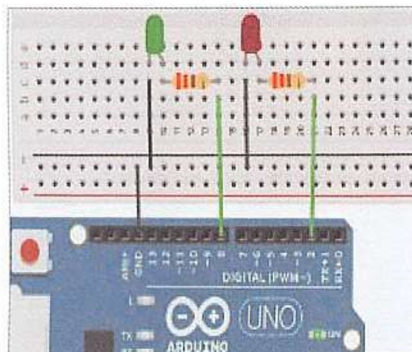


Serial Monitor

```
What color do you want to blink up, red or green?
What color do you want to blink up, red or green?
```

green

5. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำอื่นๆ ที่ไม่ใช่ red หรือ green แล้วคลิกปุ่ม Send

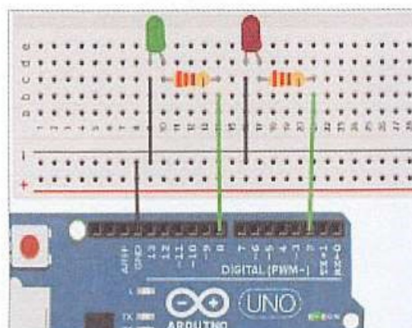


Serial Monitor

```
What color do you want to blink up, red or green?
What color do you want to blink up, red or green?
What color do you want to blink up, red or green?
```

yellow

6. เมื่อคลิกปุ่ม Send หลอดไฟ LED ทั้งสองจะไม่กะพริบ โปรแกรมจะแสดงข้อความ Invalid color choice, please try again!!! และแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง



Serial Monitor

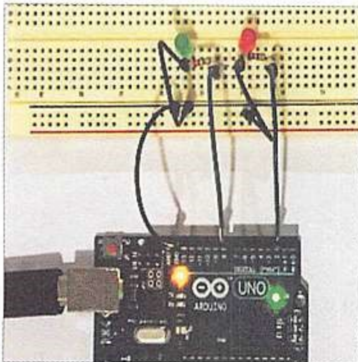
```
What color do you want to blink up, red or green?
What color do you want to blink up, red or green?
What color do you want to blink up, red or green?
Invalid color choice, please try again!!!

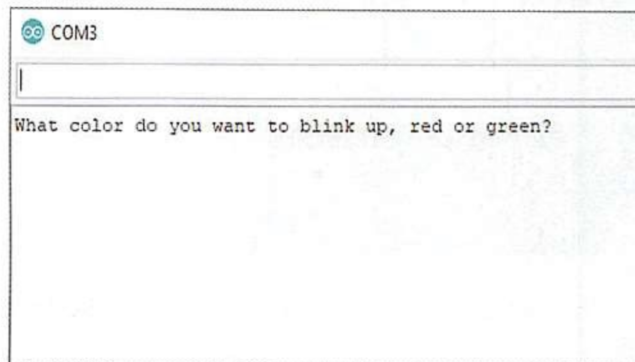
What color do you want to blink up, red or green?
```

Arduino Board Output

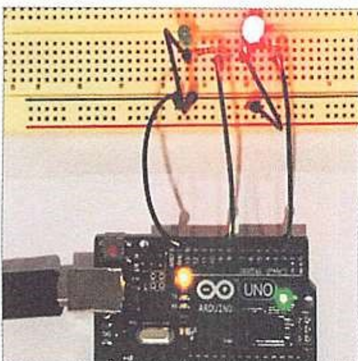
Arduino Tutorial 15 แบบ if Statement และแบบ if/else Statement ผลลัพธ์จะเหมือนกัน

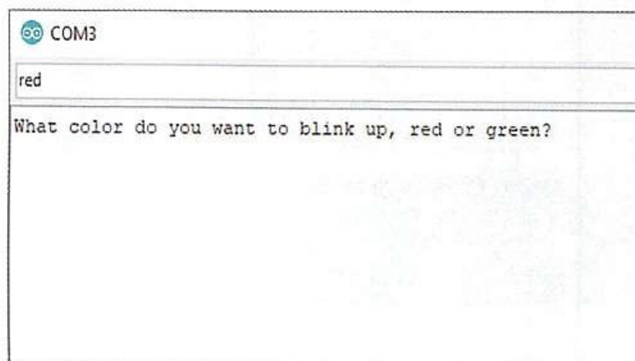
1. เมื่อรันผล Serial Monitor จะแสดงข้อความ What color do you want to blink up, red or green?



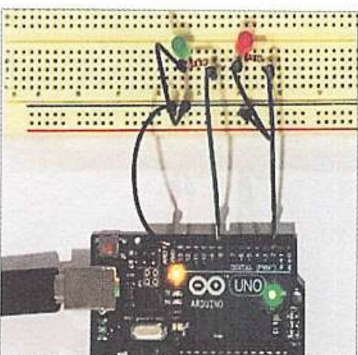


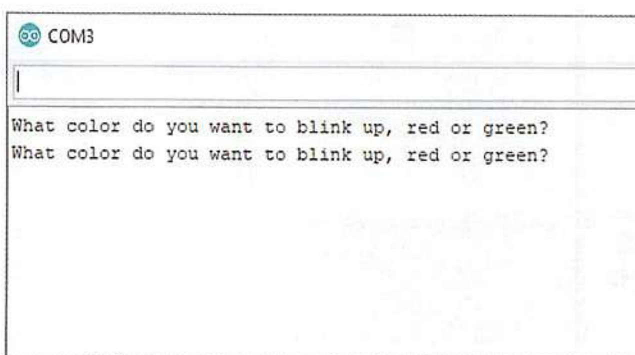
2. ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำว่า red แล้วคลิกปุ่ม Send หลอดไฟ LED สีแดงจะกะพริบ



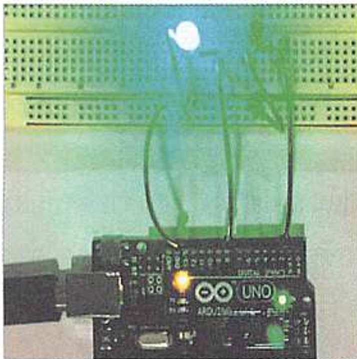


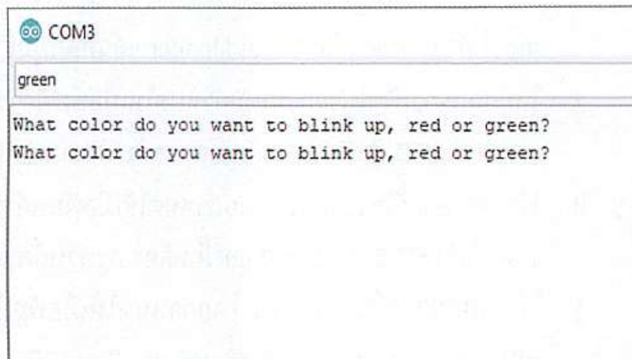
3. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง



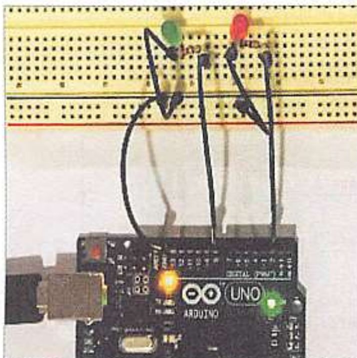


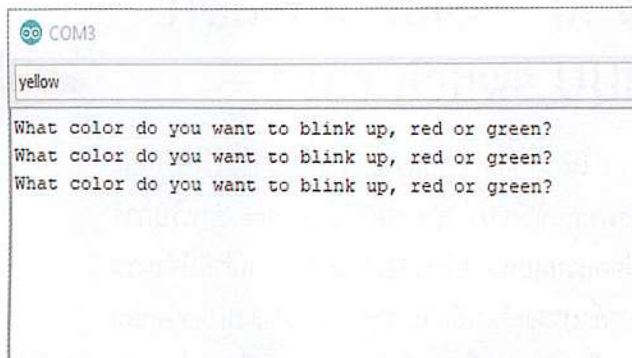
4. ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำว่า green แล้วคลิกปุ่ม Send หลอดไฟ LED สีเขียวจะกะพริบ



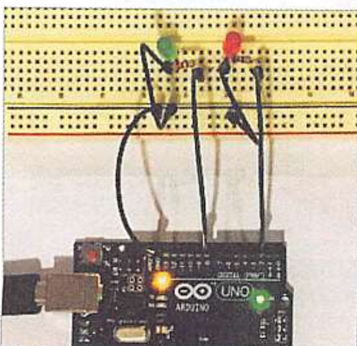


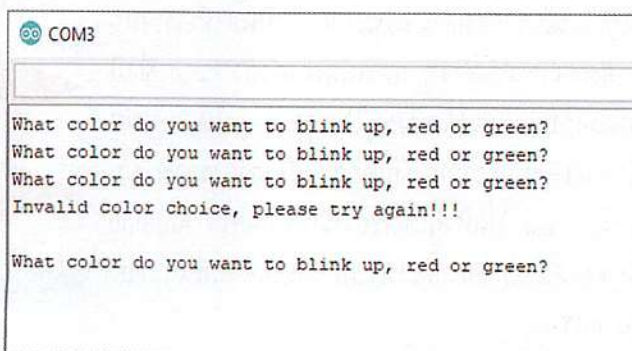
5. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง ทดลองป้อน Input ในช่อง Serial Monitor พิมพ์คำอื่นๆ ที่ไม่ใช่ red หรือ green แล้วคลิกปุ่ม Send





6. เมื่อคลิกปุ่ม Send หลอดไฟ LED ทั้งสองจะไม่กะพริบ โปรแกรมจะแสดงข้อความ Invalid color choice, please try again!!! และแสดงข้อความ What color do you want to blink up, red or green? อีกครั้ง



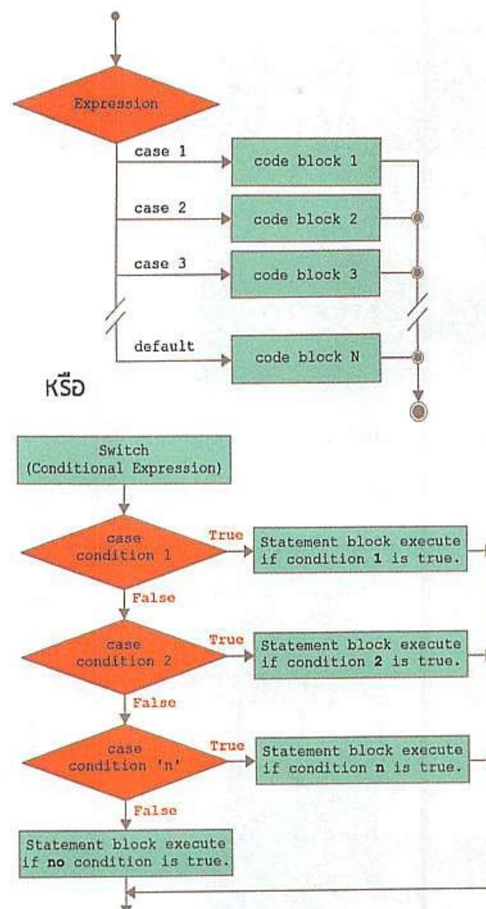


Step 10 : คำถามท้าย Arduino Tutorial 15

1. ให้นักศึกษาทดลองเพิ่มหลอดไฟ LED อีก 1 ดวง เช่น LED สีเหลือง โดยกำหนดให้เมื่อส่ง Input คำว่า yellow ใน Serial Monitor แล้วคลิกปุ่มส่ง หลอดไฟ LED สีเหลืองจะกะพริบ
2. ให้นักศึกษาแก้ไขโปรแกรมโดยกำหนดให้เมื่อพิมพ์คำว่า red ใน Serial Monitor แล้วคลิกปุ่มส่ง หลอดไฟ LED สีแดงจะกะพริบติดและดับจำนวน 5 ครั้ง
3. ให้นักศึกษาแก้ไขโปรแกรม โดยกำหนดให้เมื่อพิมพ์คำว่า all ใน Serial Monitor แล้วคลิกปุ่มส่ง หลอดไฟ LED สีแดง สีเขียว และสีเหลือง กะพริบติดและดับจำนวน 5 ครั้งพร้อมกันทั้ง 3 ดวง
4. ให้นักศึกษาแก้ไขโปรแกรม โดยกำหนดให้เมื่อพิมพ์คำว่า mix ใน Serial Monitor แล้วคลิกปุ่มส่ง หลอดไฟ LED สีแดงจะกะพริบติดและดับจำนวน 1 ครั้ง หลังจากนั้นหลอดไฟ LED สีเขียวจะกะพริบติดและดับจำนวน 2 ครั้ง และหลังจากนั้นหลอดไฟ LED สีเหลืองจะกะพริบติดและดับจำนวน 3 ครั้ง ตามลำดับ

6.10 การใช้คำสั่งเลือกแบบ switch case

ใน Arduino Tutorial ที่แล้วเราได้เรียนรู้และทำความเข้าใจการใช้งานคำสั่ง if/else สำหรับการเลือกควบคุมหลอดไฟ LED ถ้าตัวเลือกเหล่านี้มีจำนวนมากขึ้น เราจะต้องเขียนโปรแกรมที่ใช้คำสั่ง if Statement มากขึ้น อีกทางเลือกหนึ่งที่ช่วยให้การเขียนโปรแกรมง่ายขึ้น เราสามารถใช้คำสั่ง switch case มาช่วยในการควบคุมแทน if Statement ได้ โดยคำสั่ง switch จะสามารถควบคุมการทำงานของโปรแกรมโดยอนุญาตให้ผู้เขียนโปรแกรมสามารถระบุตัวแปรโค้ด เพื่อเรียกใช้คำสั่งในเงื่อนไขต่างๆ และเมื่อพบคำสั่ง case ซึ่งมีค่าตรงกับตัวแปรโค้ดที่ระบุไว้ จะควบคุมให้คำสั่งนั้นทำงาน และหลังจากคำสั่งนั้นถูกดำเนินการ เราจะเขียนคำสั่ง break ปิดท้ายแต่ละกรณีเพื่อบอกให้ Arduino หยุดคำสั่ง switch และให้ไปทำงานในส่วนอื่นๆ ของโปรแกรม



Syntax : การใช้งานคำสั่ง Switch Statement

```
switch(var) {              //var  รองรับเฉพาะ int และ char
  case label1:
    // Statements;
    break; /* optional */
  case label2:
    // Statements;
    break; /* optional */
  ...
  case labelN:
    // Statements;
    break; /* optional */

  default : /* Optional */
  statement(s);
}
```

Parameters : **var** : ตัวแปรที่มีค่าที่จะเปรียบเทียบกับกรณีต่างๆ ซึ่งมี Data Types เป็น int หรือ char

label1, label2 ... labelN : ค่าคงที่ ซึ่งมี Data Types เป็น int หรือ char

Arduino Tutorial 16 ทดลองใช้คำสั่ง switch เพื่อสั่งหลอดไฟ LED ที่ต้องการให้กะพริบ

จาก Arduino Tutorial 15 ที่ผ่านมาเราได้ใช้คำสั่ง if Statement ในการควบคุมไฟกะพริบ อีกทางเลือกหนึ่งเราสามารถใช้คำสั่ง switch Statement มาช่วยในการควบคุมไฟกะพริบได้เช่นเดียวกัน เหมาะกับการเขียนโปรแกรมแบบเลือกการควบคุมคำสั่ง หรืออุปกรณ์ให้ทำงานตามที่ต้องการได้ โดยใน Arduino Tutorial 16 นี้เราจะเพิ่มหลอดไฟ LED อีก 2 ดวง และใช้คำสั่ง switch Statement ในการควบคุม ซึ่งเป็นวิธีการที่ง่ายและสะดวก เปรียบเสมือนกับการกดปุ่มรีโมทเพื่อเลือกใช้งานอุปกรณ์ต่างๆ

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino	1	บอร์ด
2. หลอดไฟ LED	4	ดวง
3. ตัวต้านทาน 220 โอห์ม	4	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay(), Serial.read()
Switch Statement: switch...case, break

รายละเอียดการใช้งานคำสั่ง

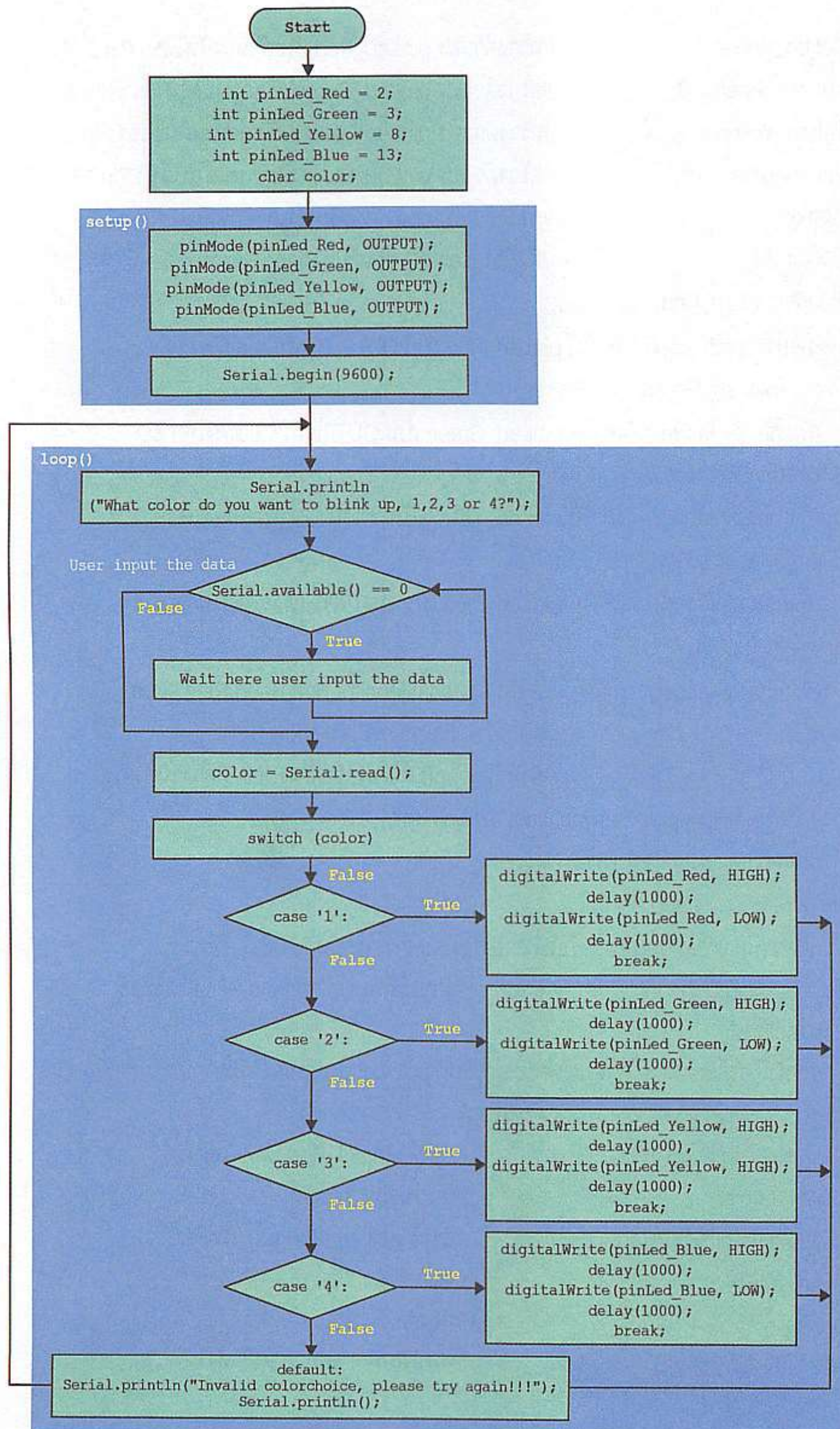
Serial.read() คำสั่งที่ใช้อ่านค่าข้อมูลอนุกรมที่รับเข้ามา เช่น เมื่อพิมพ์ค่าลงบน Serial Monitor แล้ว โปรแกรมจะประมวลผลอ่านค่าที่รับเข้ามา

Syntax : Serial.read()

Parameters : **serial** : พอร์ตอนุกรม ดูรายการพอร์ตอนุกรมของบอร์ดแต่ละรุ่นได้จากเว็บไซต์ https://www.arduino.cc/reference/en/language/functions/communication/serial/

ในการทดลองนี้จะใช้ LED 4 สี ได้แก่ **สีแดง สีเขียว สีเหลือง สีฟ้า** จึงตั้งชื่อตัวแปรเป็น **pinLed_Red**, **pinLed_Green**, pinLed_Yellow, pinLed_Blue เพื่อให้อ่านแล้วเข้าใจ หรือผู้ทดลองสามารถใช้ LED สีเดียวกันโดยแทนชื่อตัวแปรเป็น pinLed1, pinLed2, pinLed3, ..., pinLedN ตามจำนวนหลอดไฟ LED ก็ได้เช่นเดียวกัน หรือสามารถใช้ชื่อตัวแปรอื่นๆ ที่เหมาะสมได้ คือ ปรับเปลี่ยนชื่อให้เข้ากับการทดลองของตัวเองได้เลย

Step 3 : Flowchart Arduino Tutorial 16



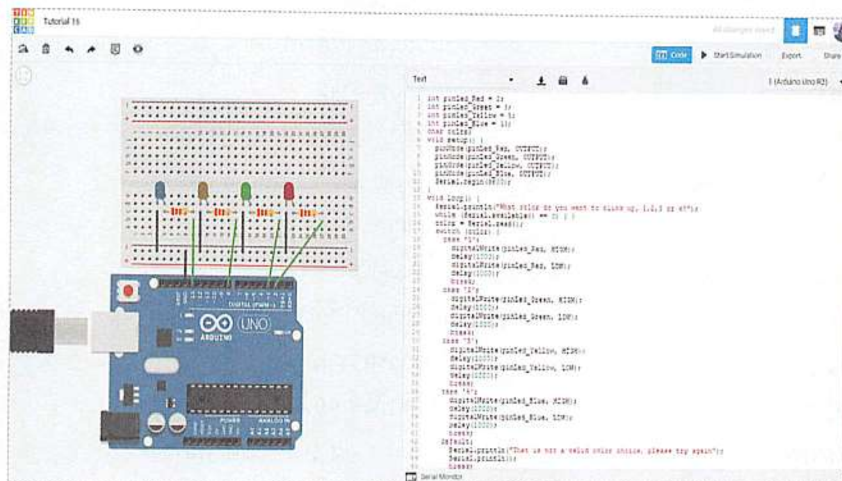
Step 4 : Source Code Arduino_Tutorial_16.ino

```
int pinLed_Red = 2;              //สร้างตัวแปร pinLed_Red เพื่อกำหนดให้ใช้ขา Pin 2
int pinLed_Green = 3;            //สร้างตัวแปร pinLed_Green เพื่อกำหนดให้ใช้ขา Pin 3
int pinLed_Yellow = 8;           //สร้างตัวแปร pinLed_Yellow เพื่อกำหนดให้ใช้ขา Pin 8
int pinLed_Blue = 13;            //สร้างตัวแปร pinLed_Blue เพื่อกำหนดให้ใช้ขา Pin 13
char color;                      //สร้างตัวแปร color สำหรับเก็บตัวอักขระ
void setup() {                   //ฟังก์ชัน setup() สำหรับตั้งค่าการใช้งานอุปกรณ์
 pinMode(pinLed_Red, OUTPUT);
     //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Red เป็น OUTPUT เพื่อแสดงไฟ LED
 pinMode(pinLed_Green, OUTPUT);
     //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Green เป็น OUTPUT เพื่อแสดงไฟ LED
 pinMode(pinLed_Yellow, OUTPUT);
     //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Yellow เป็น OUTPUT เพื่อแสดงไฟ LED
 pinMode(pinLed_Blue, OUTPUT);
     //ฟังก์ชัน pinMode() ตั้งค่าให้ pinLed_Blue เป็น OUTPUT เพื่อแสดงไฟ LED
 Serial.begin(9600);
     //ฟังก์ชัน Serial.begin สำหรับเปิดใช้งาน Serial Monitor ด้วย Baud Rate 9600
}                                //จบการทำงานของฟังก์ชัน setup()
void loop() {                    //ฟังก์ชัน loop() สำหรับสั่งให้ Arduino ทำงานที่ต้องการ
 Serial.println("What color do you want to blink up, 1,2,3 or 4?");
     //แสดงข้อความและขึ้นบรรทัดใหม่
 while (Serial.available() == 0) { }
     //เช็คว่ามีการป้อนข้อมูลมาหรือไม่ ถ้ายังไม่มีจะค้างไว้รอจนกว่าจะมี
 color = Serial.read();          //อ่านค่าตัวอักขระที่พิมพ์เข้ามาจาก Serial Monitor
 switch (color) {
     //คำสั่งเงื่อนไข switch กำหนดให้ color เป็นตัวแปรในการควบคุมให้ทำงานตามคำสั่ง
  case '1':                      //กรณีพิมพ์ 1 จะทำตามคำสั่งในนี้
   digitalWrite(pinLed_Red, HIGH);        //สั่งให้ pinLed_Red แสดงไฟติด
   delay(1000);                           //หน่วงเวลา 1 วินาที
   digitalWrite(pinLed_Red, LOW);         //สั่งให้ pinLed_Red แสดงไฟติด
   delay(1000);                           //หน่วงเวลา 1 วินาที
   break;                                 //หยุดการทำงาน
  case '2':                               //กรณีพิมพ์ 2 จะทำตามคำสั่งในนี้
```

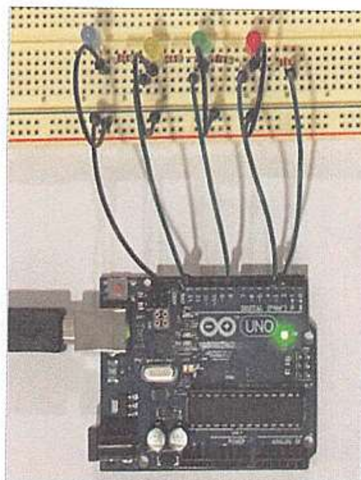
```
      digitalWrite(pinLed_Green, HIGH);        //สั่งให้ pinLed_Green แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      digitalWrite(pinLed_Green, LOW);         //สั่งให้ pinLed_Green แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      break;                                   //หยุดการทำงาน
    case '3':                                  //กรณีพิมพ์ 3 จะทำตามคำสั่งในนี้
      digitalWrite(pinLed_Yellow, HIGH);       //สั่งให้ pinLed_Yellow แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      digitalWrite(pinLed_Yellow, LOW);        //สั่งให้ pinLed_Yellow แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      break;                                   //หยุดการทำงาน
    case '4':                                  //กรณีพิมพ์ 4 จะทำตามคำสั่งในนี้
      digitalWrite(pinLed_Blue, HIGH);         //สั่งให้ pinLed_Blue แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      digitalWrite(pinLed_Blue, LOW);          //สั่งให้ pinLed_Blue แสดงไฟติด
      delay(1000);                             //หน่วงเวลา 1 วินาที
      break;                                   //หยุดการทำงาน
    default:                                   //กรณีอื่น ๆ ที่ไม่ใช่การพิมพ์ 1, 2, 3, 4
      Serial.println("Invalid color choice, please try again!!!"); //แสดงข้อความและขึ้นบรรทัดใหม่
      Serial.println();                        //ขึ้นบรรทัดใหม่
  }
}
```

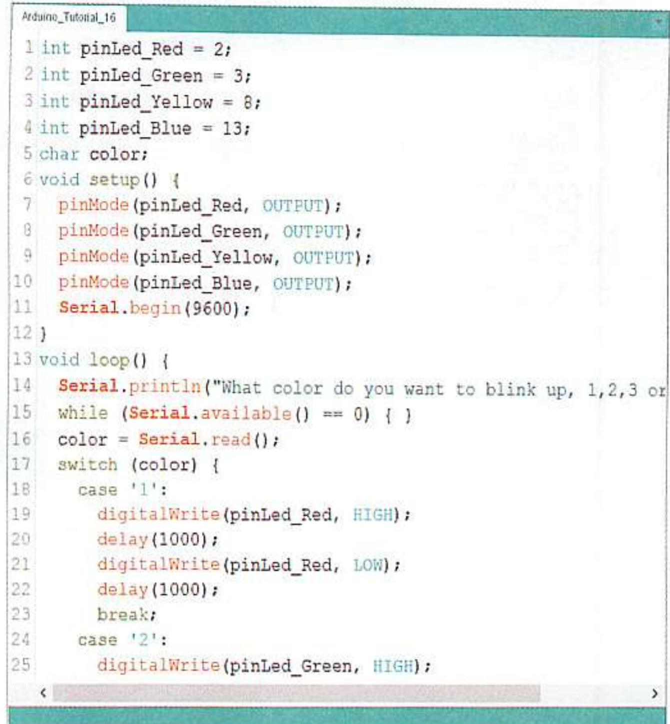
Step 5 : วิธีการทดลอง Arduino Tutorial 16

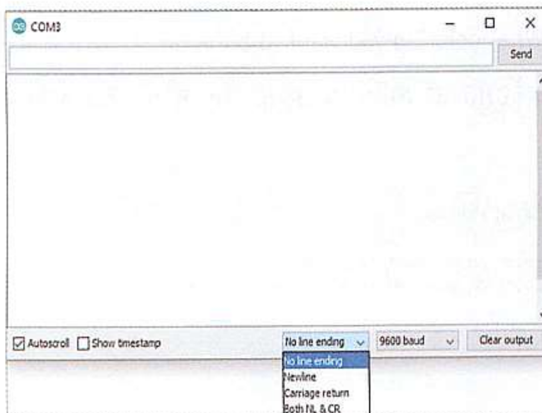
Tinkercad



Arduino Board





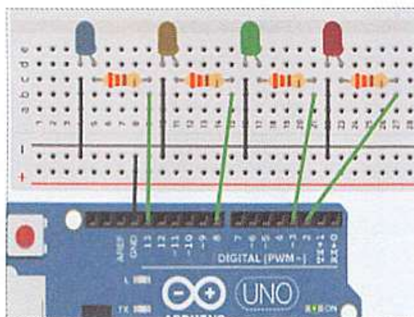


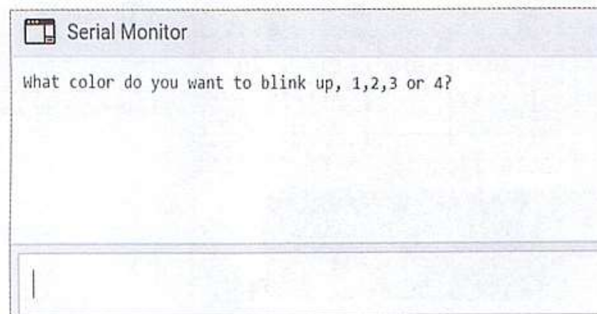
ในโปรแกรม Arduino IDE ให้เปิด Serial Monitor และปรับให้เป็น No line ending เพื่อส่งเฉพาะข้อมูลให้ไปยังคำสั่ง **Serial**.read() เพื่ออ่าน Input ที่ป้อนได้อย่างถูกต้อง

Step 6 : ผลการทดลอง Arduino Tutorial 16

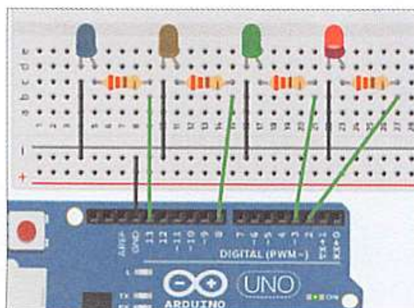
Tinkercad Output

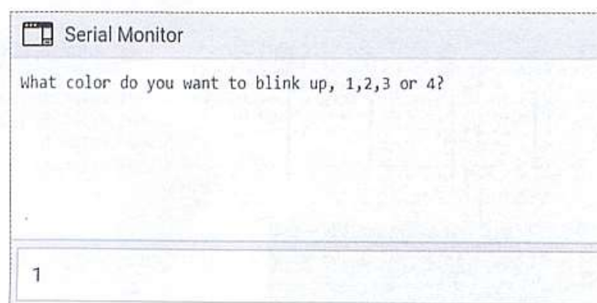
1. เมื่อรันผล Serial Monitor จะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4?



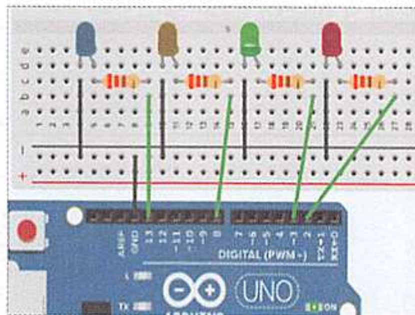


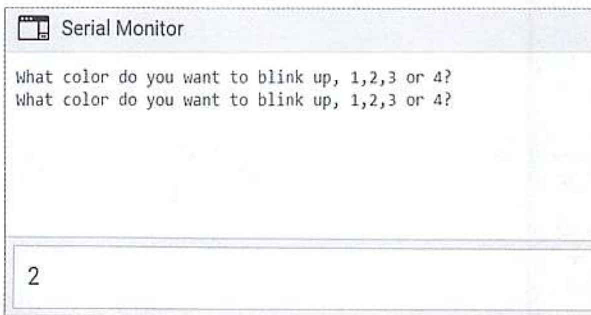
2. ในช่องป้อนข้อมูล Input ทดลองพิมพ์เลข 1 แล้วคลิกปุ่ม Send หลอดไฟ **LED สีแดง**จะกะพริบ 1 วินาที



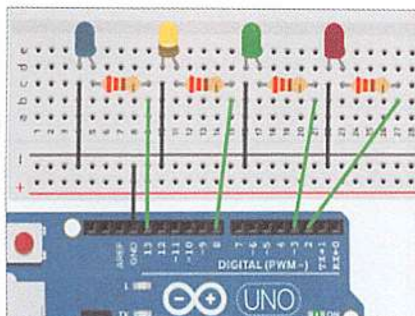


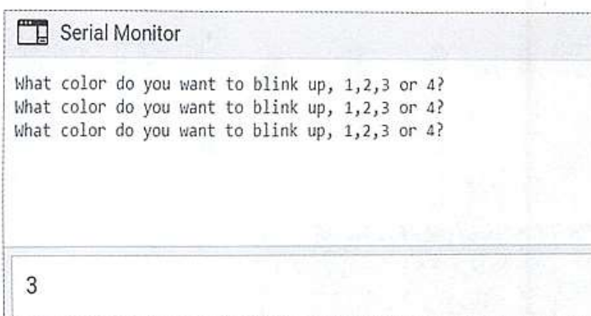
3. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 2 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ **LED สีเขียว** จะกะพริบ 1 วินาที



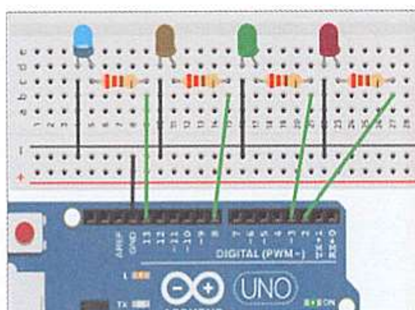


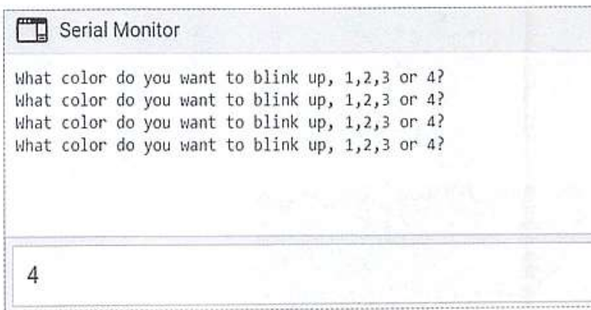
4. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 3 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ LED สีเหลือง จะกะพริบ 1 วินาที



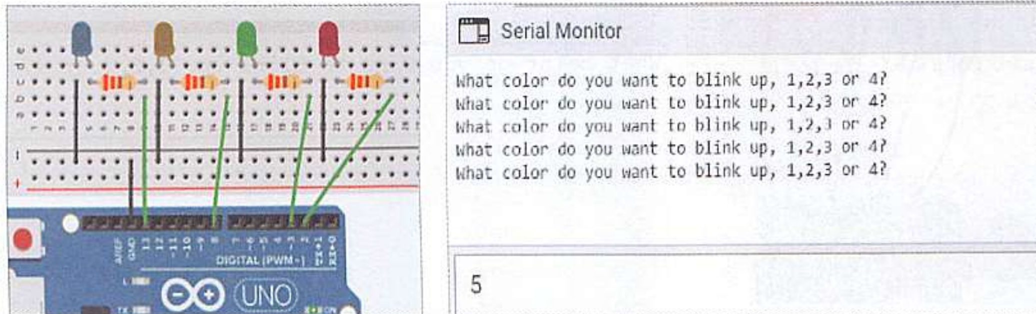


5. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 4 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ LED สีฟ้า จะกะพริบ 1 วินาที

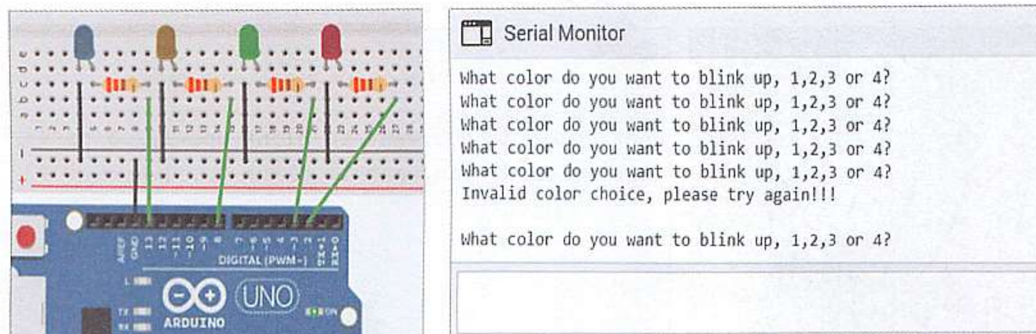




6. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 5 หรือข้อความอื่นๆ ที่ไม่ใช่เลข 1, 2, 3, 4 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send

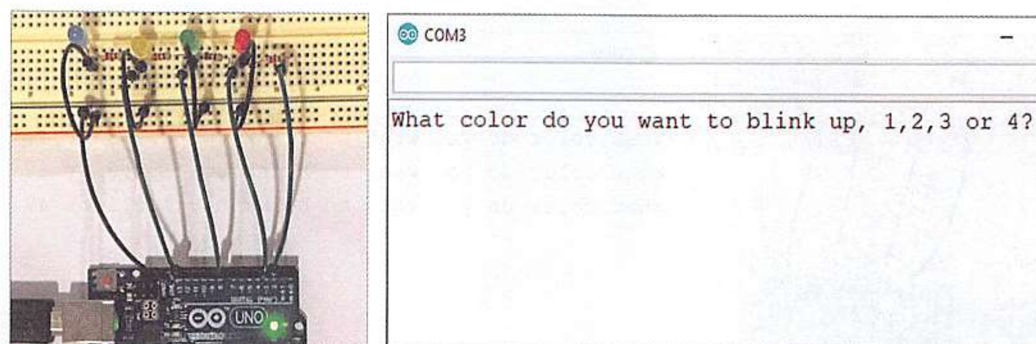


7. โปรแกรมจะแสดงข้อความ Invalid color choice, please try again!!! แล้วขึ้นบรรทัดใหม่พร้อมกับแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4?

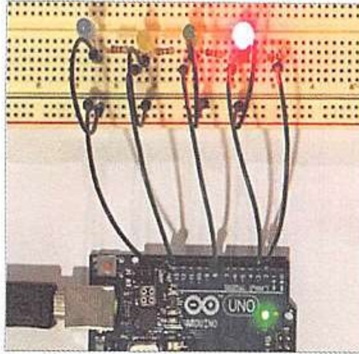


Arduino Board Output

1. เมื่อรันผล Serial Monitor จะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4?

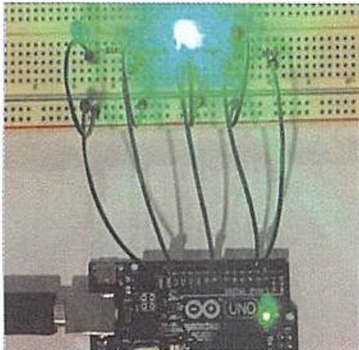


2. ในช่องป้อนข้อมูล Input ทดลองพิมพ์เลข 1 แล้วคลิกปุ่ม Send หลอดไฟ LED สีแดงจะกะพริบ 1 วินาที



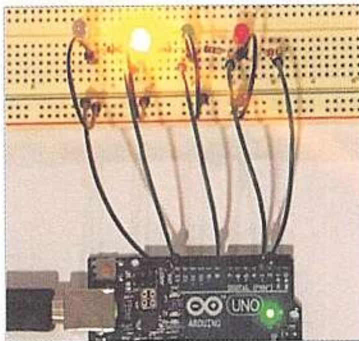

```
COM3
1
What color do you want to blink up, 1,2,3 or 4?
```

3. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 2 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ LED สีเขียวจะกะพริบ 1 วินาที



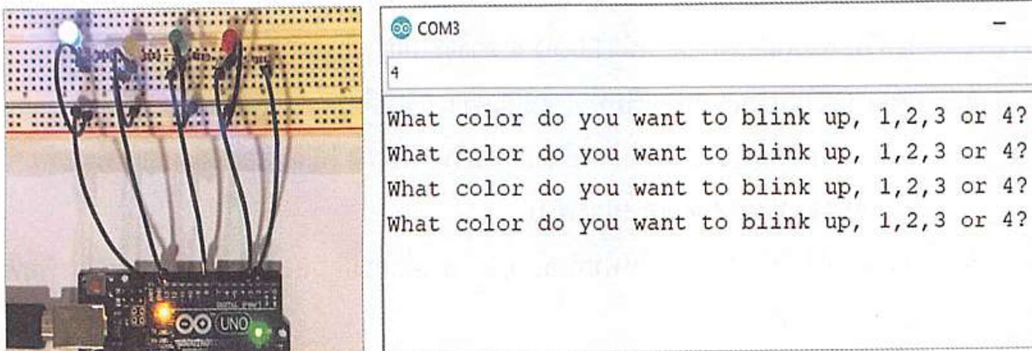

```
COM3
2
What color do you want to blink up, 1,2,3 or 4?
What color do you want to blink up, 1,2,3 or 4?
```

4. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 3 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ LED สีเหลืองจะกะพริบ 1 วินาที

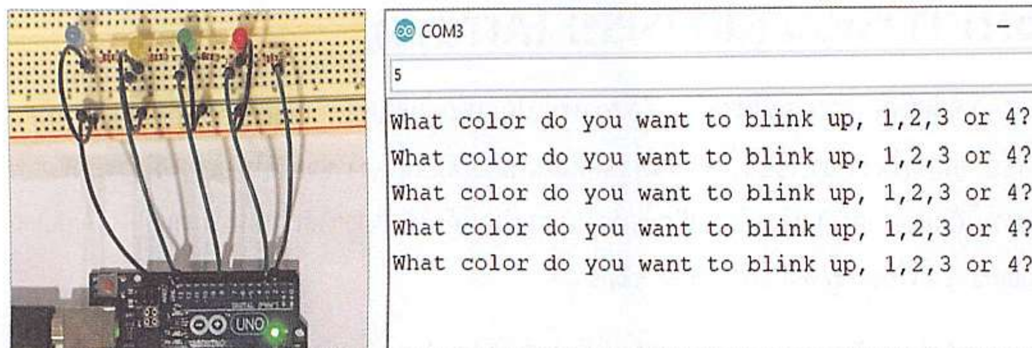



```
COM3
3
What color do you want to blink up, 1,2,3 or 4?
What color do you want to blink up, 1,2,3 or 4?
What color do you want to blink up, 1,2,3 or 4?
```

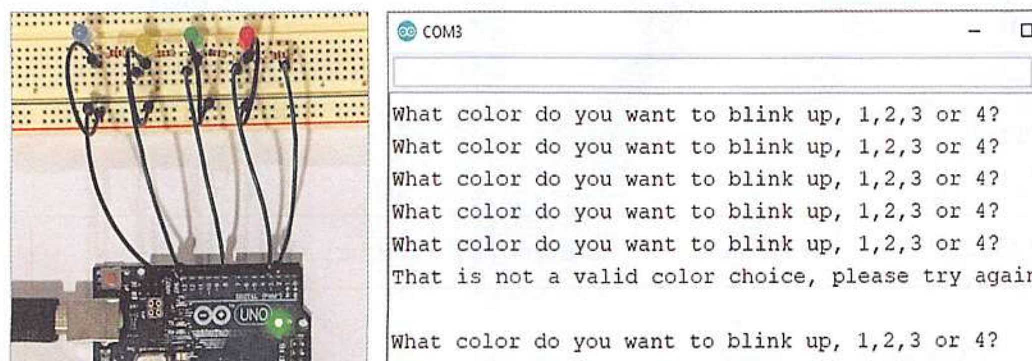
5. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 4 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send หลอดไฟ LED สีฟ้าจะกะพริบ 1 วินาที



6. หลังจากนั้นโปรแกรมจะแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4? อีกครั้ง ทดลองพิมพ์เลข 5 หรือข้อความอื่นๆ ที่ไม่ใช่เลข 1, 2, 3, 4 ในช่องป้อนข้อมูล แล้วคลิกปุ่ม Send



7. โปรแกรมจะแสดงข้อความ That is not a valid color choice, please try again แล้วขึ้นบรรทัดใหม่ พร้อมกับแสดงข้อความ What color do you want to blink up, 1, 2, 3 or 4?



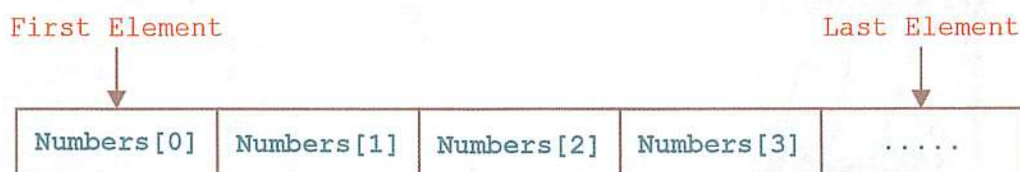
Step 7 : คำถามท้าย Arduino Tutorial 16

1. ทดลองแก้ไขโปรแกรมโดยพิมพ์ 1 LED สีฟ้า (LED4) กะพริบ, พิมพ์ 2 LED สีเหลือง (LED3) กะพริบ, พิมพ์ 3 LED สีเขียว (LED2) กะพริบ, พิมพ์ 4 LED สีแดง (LED1) กะพริบ พิมพ์นอกเหนือจากนี้ LED ทั้ง 4 ดวงจะไม่กะพริบ
2. ทดลองแก้ไขโปรแกรมโดยพิมพ์ 1 ให้สีแดงและสีเหลืองกะพริบ, พิมพ์ 2 ให้สีเขียวและสีฟ้ากะพริบ, พิมพ์ 3 ให้สีแดงและสีเขียวกะพริบ, พิมพ์ 4 ให้สีเหลืองและสีฟ้ากะพริบ พิมพ์นอกเหนือจากนี้ทั้ง 4 ดวงจะไม่กะพริบ
3. ทดลองแก้ไขโปรแกรมโดยให้พิมพ์ a, b, c, d เพื่อควบคุมหลอดไฟ LED แทนการพิมพ์ 1, 2, 3, 4 ตามลำดับ
4. ทดลองแก้ไขโปรแกรมโดยให้พิมพ์ a ควบคุมให้ไฟวิ่งจาก LED1 ไป LED4, พิมพ์ b ควบคุมให้ไฟวิ่งจาก LED4 ไป LED1, พิมพ์ c ควบคุมให้ LED1 ไป LED4 และกลับมาที่ LED1 (วิ่งไปวิ่งกลับ) พิมพ์ d ให้ LED ทั้ง 4 ดวงกะพริบพร้อมกัน

6.11 การใช้งานอาร์เรย์ (Arrays)

เทคนิคที่ดีในการเขียนโปรแกรม คือ การทำให้โค้ดเพียงไม่กี่บรรทัดแต่สามารถทำงานได้มากที่สุด หรืออาจกล่าวได้ว่า การเขียนโค้ดที่สั้นที่สุดแต่มีพลังมากที่สุดนั่นเอง ช่วยให้โปรแกรมมีประสิทธิภาพทำงานได้อย่างรวดเร็ว และขจัดข้อผิดพลาดในการเขียนโค้ดได้ ซึ่งเราได้เริ่มต้นทำมาแล้วใน Arduino Tutorial 14 เรื่อง for/while/do while loops

ใน Arduino Tutorial นี้ เราจะมาเรียนรู้เรื่องการใช้งาน Array ซึ่งเป็นชุดของตัวแปรที่มีการจัดเก็บข้อมูลแบบกลุ่มที่เป็นชนิดข้อมูลเดียวกัน ในการใช้งาน Array เราสามารถเข้าถึงได้ด้วยหมายเลข Index ตัวแปรแต่ละตัวใน Array เราจะเรียกว่า "Element" โดยมีวิธีการประกาศตัวแปรและเรียกใช้งานตัวแปรดังนี้



การใช้งานตัวแปร Array จะประกอบด้วยการประกาศตัวแปรและเรียกใช้งานตัวแปร Array

Syntax การประกาศตัวแปร Array

- **Single-Dimensional Arrays (One Dimensional Arrays)**

```
DataType arrayName[size];
DataType arrayName[size] = {element1, element2, ..., elementN};
DataType arrayName[size] = "Text Message";

เช่น

int myInts[9];
int myPins[5] = {2, 4, 6, 8, 13};
int mySensValues[6] = {12, -23, 99, -7, 23, 9};
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
char myMessage[13] = "Hello Arduino";
```

- **Multi-Dimensional Arrays (≥ Two Dimensional Arrays)**

```
DataType arrayName[size1][size2]...[sizeN];

เช่น

int twoDim[3][9];
```

การเรียกใช้งานตัวแปร Array

ในการเรียกใช้งานตัวแปร Array ตำแหน่งที่เก็บค่าข้อมูลจะเรียกว่า Index โดยเริ่มตั้งแต่ Index ตำแหน่งที่ 0, 1, 2, ..., N

	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

จากรูป ตัวแปร Array ชื่อ balance เป็นชนิดข้อมูล double เก็บได้จำนวน 5 ข้อมูล ได้แก่ 1000.0, 2.0, 3.4, 7.0, 5.0 ตามลำดับ ตัวอย่างเช่น

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

ถ้าต้องการดึงข้อมูลตัวเลขในแต่ละค่าข้อมูลมาใช้งาน ต้องเรียกใช้โดยอ้างอิงตำแหน่ง Index ของ Array ได้แก่

```
balance[0] = 1000.0;
balance[1] = 2.0;
balance[2] = 3.4;
balance[3] = 7.0;
balance[4] = 50.0;
```

ดังนั้น ถ้าต้องการดึงค่าข้อมูล 3.4 และ 50.0 เราจะเรียกใช้ด้วยการเขียนคำสั่ง balance[2] และ balance[4] เป็นต้น

หรือ char myMessage[13] = "Hello Arduino";

ในกรณีที่ต้องการดึงข้อมูลตัวอักขระในแต่ละค่าข้อมูลมาใช้งาน ต้องเรียกใช้ด้วยตำแหน่งของ Index ใน Array ได้แก่

```
myMessage[0] = 'H';
myMessage[1] = 'e';
myMessage[2] = 'l';
myMessage[3] = 'l';
myMessage[4] = 'o';
myMessage[5] = ' ';
myMessage[6] = 'A';
myMessage[7] = 'r';
myMessage[8] = 'd';
myMessage[9] = 'u';
myMessage[10] = 'i';
myMessage[11] = 'n';
myMessage[12] = 'o';
```

ดังนั้น ถ้าต้องการดึงข้อมูลตัวอักขระ A และ i มาใช้งาน ต้องเรียกใช้ด้วยตำแหน่งของ Index ใน Array

```
myMessage[6] = 'A';  //เริ่มตั้งแต่ 0 โดยนับตัวอักขระทุกตัวรวมช่องว่างนับเป็นตัวอักขระ
myMessage[10] = 'i';
```

Arduino Tutorial 17 ทดลองใช้งาน Array ควบคุมหลอดไฟ LED

จากแล็บ Arduino Tutorial 16 ที่ผ่านมา เราได้เพิ่มอุปกรณ์หลอดไฟ LED รวมจำนวน 4 ดวง โดยกำหนดขา Pin ทีละ Pin ในกรณีที่เรามีจำนวนของตัวแปร หรืออุปกรณ์ต่อพ่วงต่างๆ มากขึ้น ในการกำหนดขาหรือค่าข้อมูลต่างๆ เราสามารถเขียนโปรแกรมโดยใช้ Array ซึ่งเป็นโครงสร้างข้อมูลที่สามารถเก็บ และเรียกใช้ค่าของตัวแปรแบบชุดที่เป็นชนิดเดียวได้สะดวกมากขึ้น

Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino 1 บอร์ด
2. หลอดไฟ LED 4 ดวง
3. ตัวต้านทาน 220 โอห์ม 4 ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

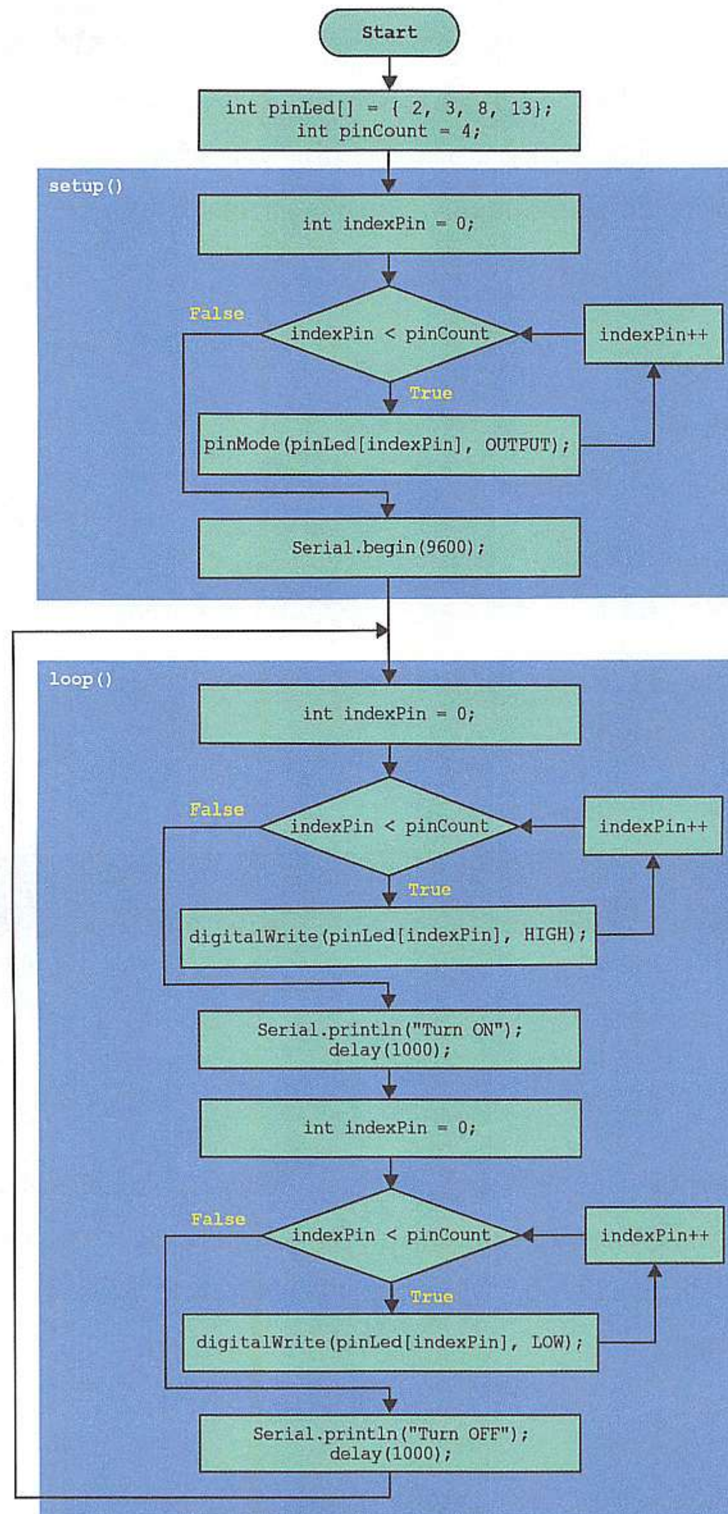
```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay(), Serial.read()
Array: int pinLed[]
```

เปรียบเทียบการประกาศค่าของตัวแปร

แบบทั่วไป	แบบใช้ Array
int pinLed_Red = 2; int pinLed_Green = 3; int pinLed_Yellow = 8; int pinLed_Blue = 10;	int pinLed[] = { 2, 3, 8, 10};

จะเห็นว่า การใช้งานแบบ Array จะเขียนโปรแกรมได้จำนวนบรรทัดที่สั้นและประหยัดเวลากว่าแบบทั่วไป โดยการใช้ Array เหมาะสมที่จะนำไปใช้ในกรณีถ้าโปรแกรมที่เราเขียนมีการกำหนดค่าหลายๆ ค่าที่เป็นชนิดข้อมูลเดียวกัน (Data Types เหมือนกัน) แนะนำให้เลือกใช้งาน Array ในการประกาศค่าของตัวแปร

Step 3 : Flowchart Arduino Tutorial 17

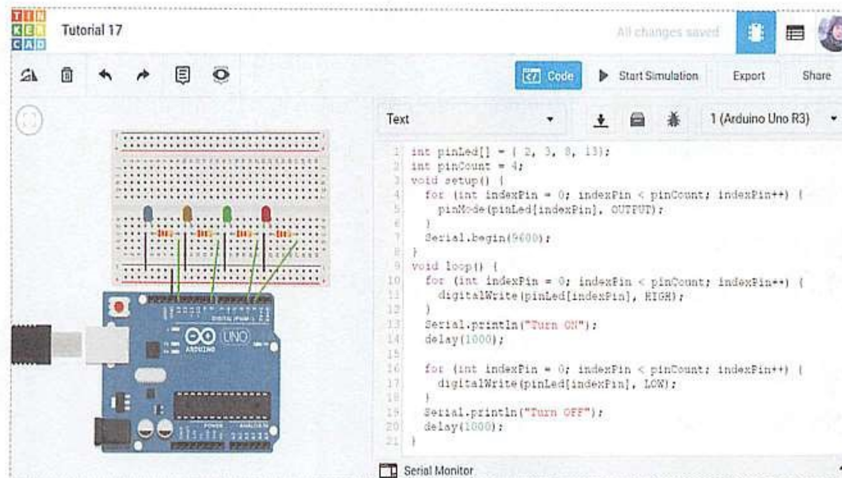


Step 4 : Source Code Arduino_Tutorial_17.ino

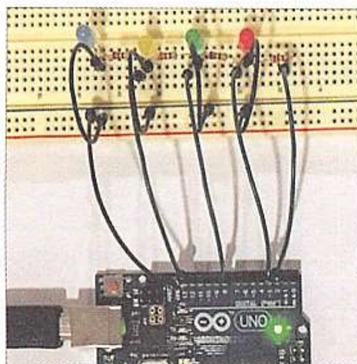
```
int pinLed[] = { 2, 3, 8, 13};
    //สร้างตัวแปร pinLed แบบ Array เพื่อกำหนดให้ใช้ขา Pin หมายเลข 2, 3, 8, 13
int pinCount = 4;                //สร้างตัวแปร pinCount เพื่อกำหนดให้การนับขา Pin ทั้ง 4 Pins
void setup() {                   //ฟังก์ชัน setup() สำหรับตั้งค่าการใช้งานอุปกรณ์
 for (int indexPin = 0; indexPin < pinCount; indexPin++) {
    //คำสั่ง for วนลูปการกำหนดหมายเลขขาทั้ง 4 Pins
  pinMode(pinLed[indexPin], OUTPUT);
    //ฟังก์ชัน pinMode() ตั้งค่าทั้ง 4 pins ให้เป็นโหมด OUTPUT
 }                               //ออกจาก for loop ที่ 1
 Serial.begin(9600);
    //ฟังก์ชัน Serial.begin สำหรับเปิดใช้งาน Serial Monitor ด้วย Baud Rate 9600
}                                //จบฟังก์ชัน setup()
void loop() {                    //ฟังก์ชัน loop() สำหรับสั่งให้ทำงานที่ต้องการ
 for (int indexPin = 0; indexPin < pinCount; indexPin++) {
    //คำสั่ง for วนลูปทีละขา Pin 2, 3, 8, 13
  digitalWrite(pinLed[indexPin], HIGH);    //สั่งให้ไฟติดทั้ง 4 LEDs
 }                                         //ออกจาก for loop ที่ 2
 Serial.println("Turn ON");                //แสดงข้อความและขึ้นบรรทัดใหม่
 delay(1000);                              //หน่วงเวลา 1 วินาที
 for (int indexPin = 0; indexPin < pinCount; indexPin++) {    //คำสั่ง for วนลูปทีละขา Pin 2, 3, 8, 13
  digitalWrite(pinLed[indexPin], LOW);     //สั่งให้ไฟดับทั้ง 4 LEDs
 }                                         //ออกจาก for loop ที่ 3
 Serial.println("Turn OFF");               //แสดงข้อความและขึ้นบรรทัดใหม่
 delay(1000);                              //หน่วงเวลา 1 วินาที
}                                          //จบฟังก์ชัน loop()
```

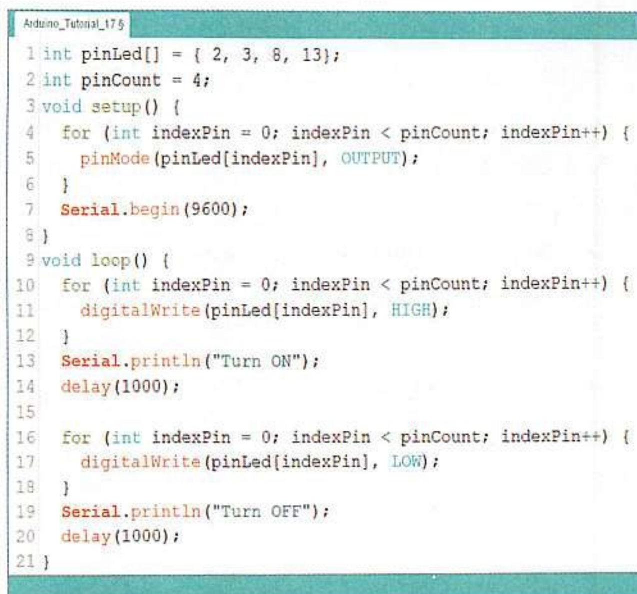
Step 5 : วิธีการทดลอง Arduino Tutorial 17

Tinkercad



Arduino Board





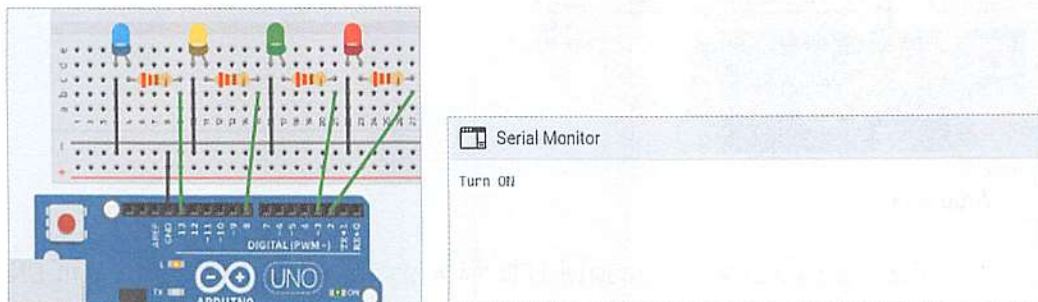

```
int pinLed[] = { 2, 3, 8, 13};
int pinCount = 4;
void setup() {
  for (int indexPin = 0; indexPin < pinCount; indexPin++) {
    pinMode(pinLed[indexPin], OUTPUT);
  }
  Serial.begin(9600);
}
void loop() {
  for (int indexPin = 0; indexPin < pinCount; indexPin++) {
    digitalWrite(pinLed[indexPin], HIGH);
  }
  Serial.println("Turn ON");
  delay(1000);

  for (int indexPin = 0; indexPin < pinCount; indexPin++) {
    digitalWrite(pinLed[indexPin], LOW);
  }
  Serial.println("Turn OFF");
  delay(1000);
}
```

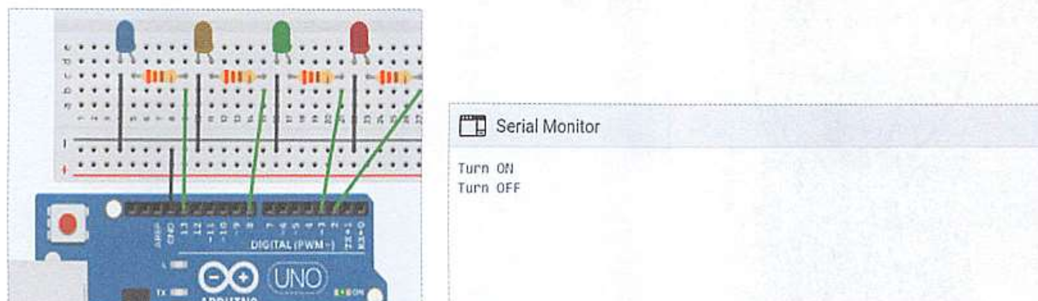
Step 6 : ผลการทดลอง Arduino Tutorial 17

Tinkercad Output

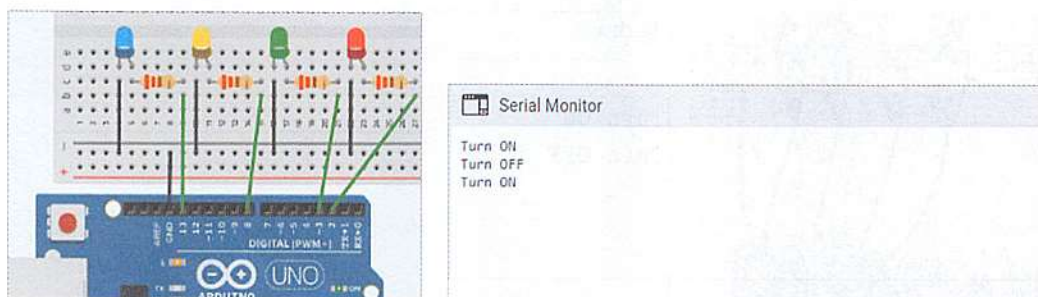
1. เมื่อรันผลโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติด พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



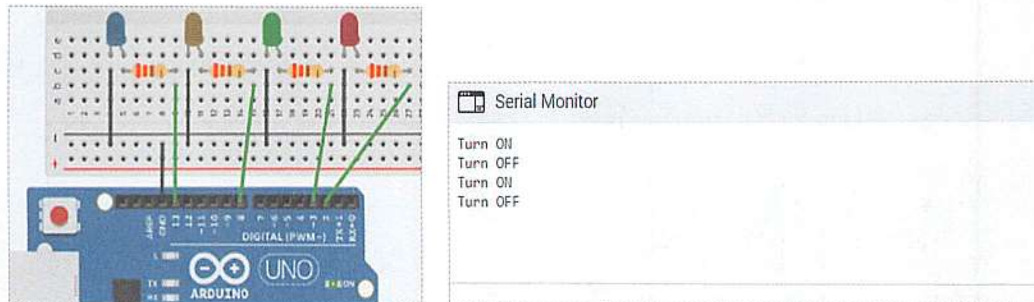
2. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับ พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที



3. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติดอีกครั้ง พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที

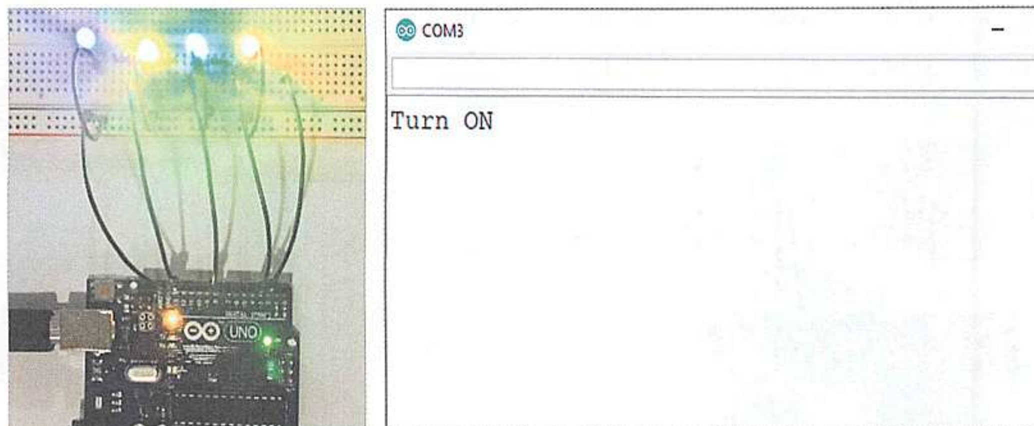


4. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับอีกครั้ง พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที และจะรันผลแบบนี้ไปเรื่อยๆ

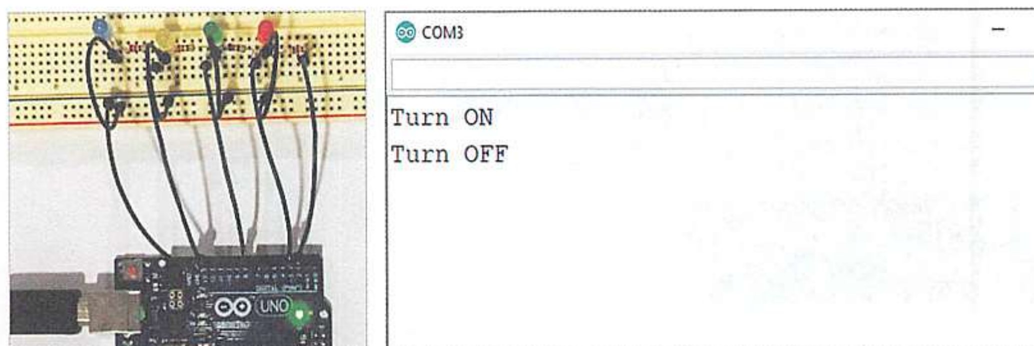


Arduino Output

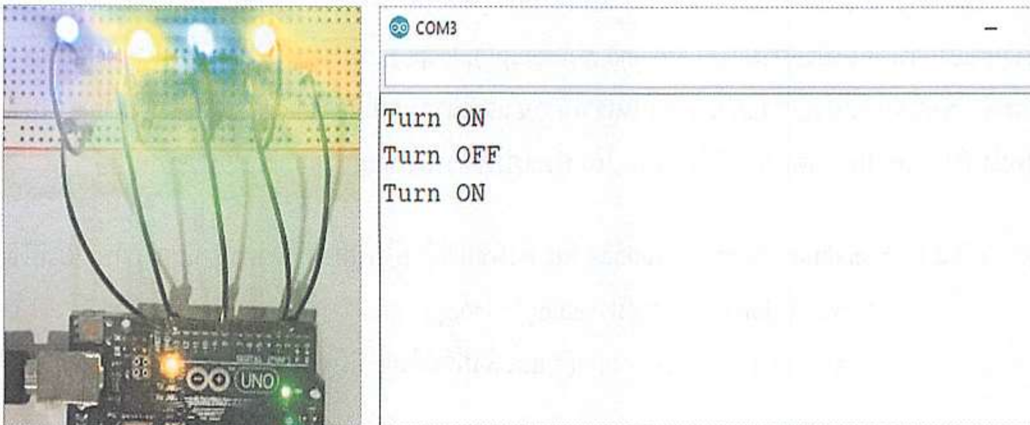
1. เมื่อรันผลโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติด พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



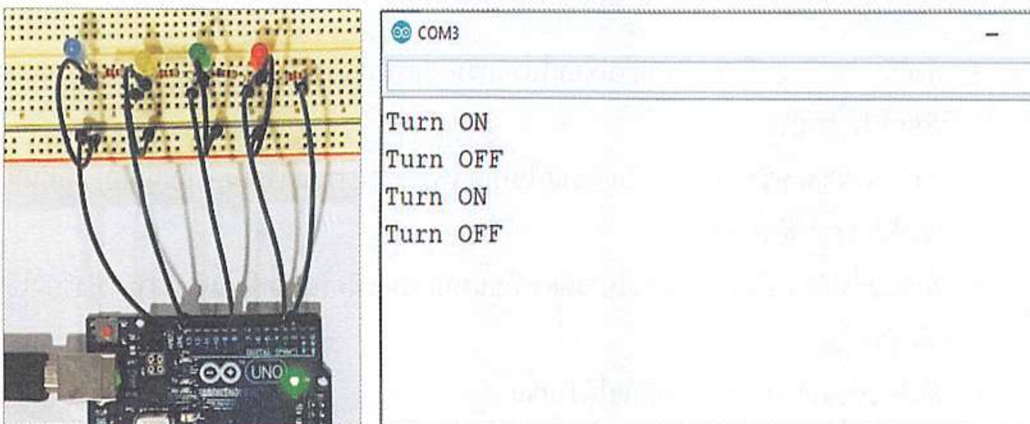
2. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับ พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที



3. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติดอีกครั้ง พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



4. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับอีกครั้ง พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที และจะรันผลแบบนี้ไปเรื่อยๆ



Step 7 : คำถามท้าย Arduino Tutorial 17

1. ตัวแปรใดคือตัวแปรชนิด Array มีกี่ตัวแปร และเก็บค่าอะไรในตัวแปรนั้น
2. ตัวแปร indexPin ใน for loop ทำหน้าที่อะไร
3. ทดลองแก้ไขโปรแกรมโดยเปลี่ยนค่า pinCount จาก 4 เป็น 3 ผลลัพธ์ที่ได้เป็นอย่างไร เพราะเหตุใด
4. ถ้าต้องการเพิ่ม LED อีก 1 ดวง รวมทั้งหมด 5 ดวง แล้วสั่งให้กะพริบติดและดับพร้อมกัน จะทำอย่างไร

6.12 การสร้าง Functions ด้วยตัวเอง

ในการเขียนโปรแกรมที่ผ่านมา เราจะเห็นว่ามีการใช้งานคำสั่งที่เป็นฟังก์ชัน (Functions) พื้นฐานต่างๆ ของการเขียนโปรแกรมแบบ Arduino เช่น setup(), loop(), pinMode(), digitalWrite(), delay() และฟังก์ชันอื่นๆ ที่โปรแกรม Arduino มีให้เราใช้งาน แต่ที่น่าสนใจก็คือ เราสามารถสร้างฟังก์ชันขึ้นใหม่เพื่อให้ทำงานตามที่เราต้องการได้เช่นกัน โดยประเภทของฟังก์ชันแบ่งออกได้ 2 ประเภท คือ

6.12.1 Standard Library Functions คือ ฟังก์ชันที่มีอยู่ในไลบรารีมาตรฐานของภาษา C ที่ใช้สำหรับ Arduino อย่างเช่น setup(), loop(), pinMode(), digitalWrite(), delay(), Serial.begin(), Serial.println() และฟังก์ชันอื่นๆ ที่มีอยู่ในโปรแกรมของ Arduino

6.12.2 User Defined Functions คือ ฟังก์ชันที่นักเขียนโปรแกรมสร้างขึ้นมาเอง โดยสร้างขึ้นได้ตามความต้องการของผู้ใช้ และความซับซ้อนของโปรแกรม

การสร้างฟังก์ชันขึ้นใช้เองได้เป็นส่วนที่สำคัญอย่างมากในการเขียนโปรแกรม เพราะว่า

- ฟังก์ชันช่วยจัดระเบียบให้กับการเขียนโปรแกรม และอาจช่วยในการกำหนดแนวทางในการเขียนโปรแกรมได้
- ฟังก์ชันช่วยลดความผิดพลาดในการแก้ไขโค้ด เพราะสามารถแก้ไขในฟังก์ชันนั้นๆ ได้ทันทีโดยไม่กระทบฟังก์ชันอื่นๆ
- ฟังก์ชันทำให้การเขียนโปรแกรมใน Sketch มีขนาดที่เล็กกะทัดรัดมากขึ้น เพราะสามารถเรียกใช้โค้ดซ้ำๆ ได้
- ฟังก์ชันช่วยให้การอ่านโค้ดเข้าใจได้ง่ายขึ้น

Syntax การสร้าง Functions

```
return_type function_name(parameter)
{
  body of the function
}
```

return_type : ชนิดข้อมูลของค่าที่ฟังก์ชันต้องการคืนค่า เช่น int function_name(); จะต้องการคืนค่า return value; บางฟังก์ชันอาจไม่ต้องการคืนค่า เช่น return_type ที่เป็น void setup(), loop()

function_name : ชื่อของฟังก์ชัน สามารถตั้งชื่อให้เหมาะสมกับงานที่เมื่ออ่านชื่อแล้วรู้เลยว่าฟังก์ชันนี้ทำหน้าที่อะไร

parameter : ตัวแปรที่รับค่าข้อมูลของฟังก์ชัน บางฟังก์ชันต้องการให้ผู้ใช้ใส่ค่าข้อมูลและส่งค่ามาที่พารามิเตอร์เพื่อนำไปคำนวณในฟังก์ชัน หรือบางฟังก์ชันจะไม่มีการรับค่าพารามิเตอร์ก็ได้ ขึ้นอยู่กับการออกแบบฟังก์ชันที่ใช้

body of the function : พื้นที่ของคำสั่งการทำงานของฟังก์ชัน

Syntax การเรียกใช้ Functions

```
return_type function_name(parameter);
```

ตัวอย่างเช่น

การสร้างฟังก์ชัน

แบบมี parameter	แบบไม่มี parameter
`int myMultiplyFunction(int x, int y){` `int result;` `result = x * y;` `return result;` `}`	`void myMultiplyFunction(){` `int result;` `result = 3 * 9;` `}`

การเรียกใช้ฟังก์ชัน

แบบมี parameter	แบบไม่มี parameter
k = myMultiplyFunction(3, 9);	k = myMultiplyFunction();

Arduino Tutorial 18 ทดลองสร้างและเรียกใช้งานฟังก์ชัน เพื่อควบคุมหลอดไฟ LED

จาก Arduino Tutorial ที่ผ่านมาจะมีคำสั่งที่ใช้ควบคุมหลอดไฟ LED ให้ติดและดับ พร้อมแสดงผลออกทาง Serial Monitor ได้ เพื่อให้การเขียนโปรแกรมเป็นระบบและอ่านเข้าใจง่ายมากขึ้นใน Arduino Tutorial 18 นี้มาช่วยกันสร้างและเรียกใช้งานฟังก์ชันเพื่อควบคุมหลอดไฟ LED ผลลัพธ์การทดลองจะเหมือนกับ Arduino Tutorial 17

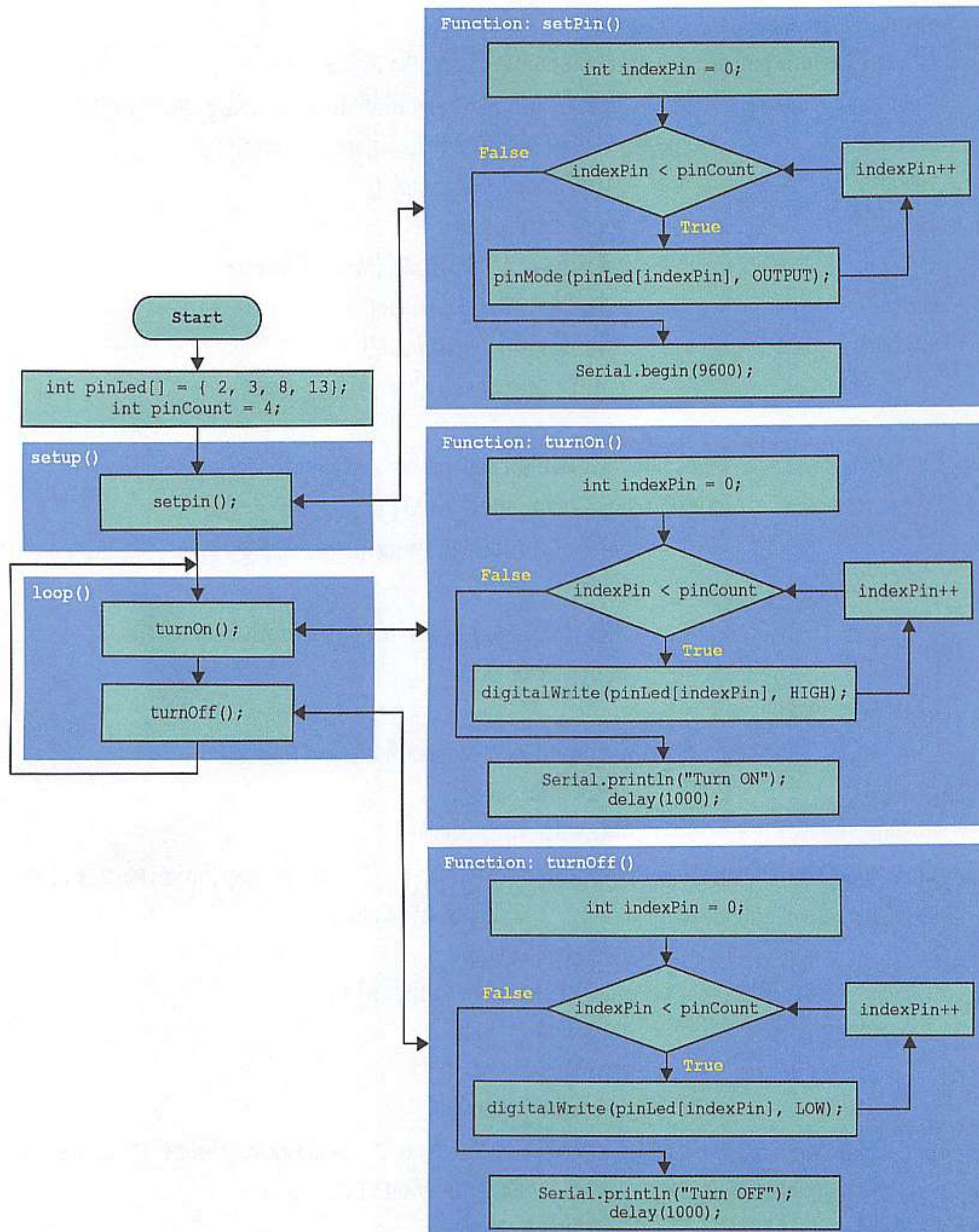
Step 1 : อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ด Arduino	1	บอร์ด
2. หลอดไฟ LED	4	ดวง
3. ตัวต้านทาน 220 โอห์ม	4	ตัว (หรือค่าที่เหมาะสม)

Step 2 : คำสั่งที่ใช้ในการทดลอง

```
setup(): pinMode(), digitalWrite(), Serial.begin()
loop(): digitalWrite(), Serial.println(), delay()
 Array: int pinLed[]
 for Statement: for(initiation; condition; increment/decrement)
 Functions ที่สร้างขึ้นเอง: setPin(), turnON(), turnOff()
```

Step 3 : Flowchart Arduino Tutorial 18

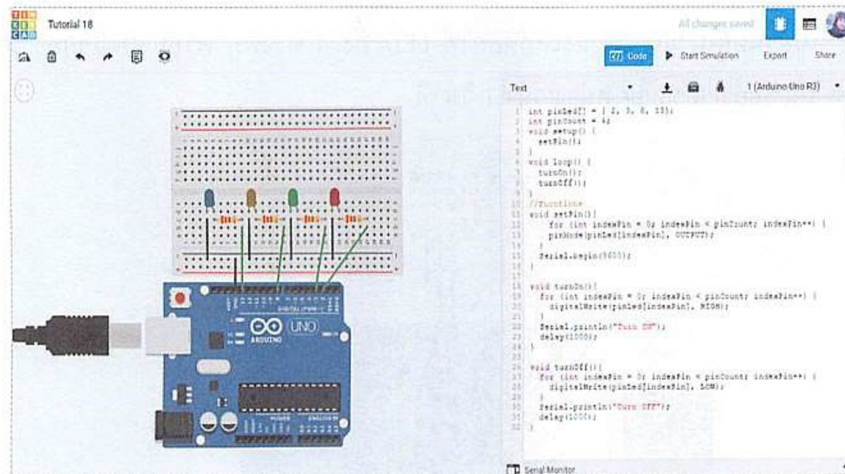


Step 4 : Source Code Arduino_Tutorial_18.ino

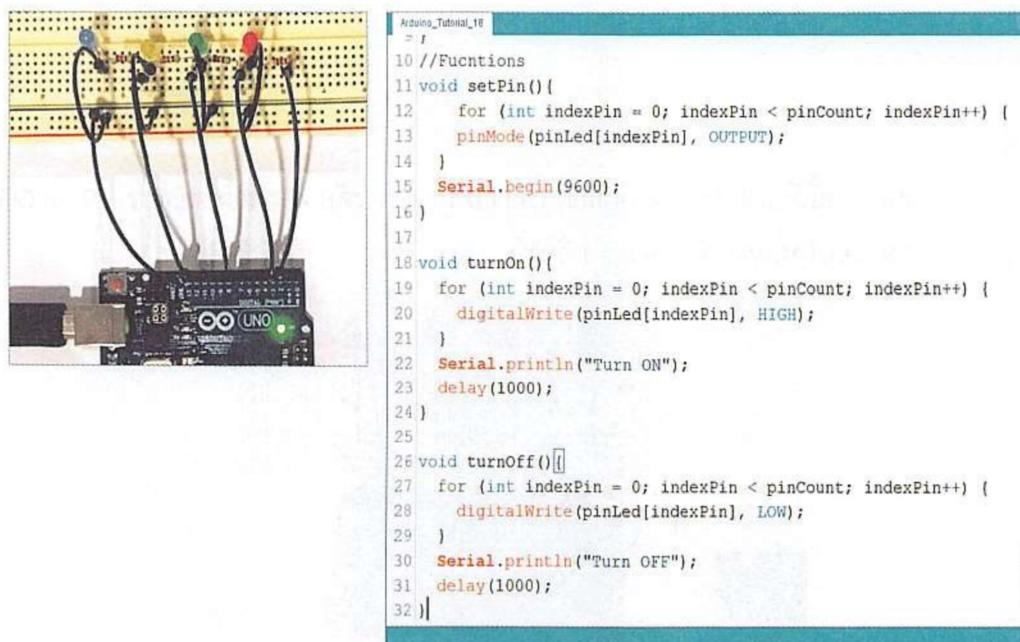
```
int pinLed[] = { 2, 3, 8, 13};
     //สร้างตัวแปร pinLed แบบ Array เพื่อกำหนดให้ใช้ขา Pin 2, 3, 8, 13
int pinCount = 4;                //สร้างตัวแปร pinCount เพื่อกำหนดให้การนับขา Pin ทั้ง 4 Pins
void setup() {                   //ฟังก์ชัน setup() สำหรับตั้งค่าการใช้งานอุปกรณ์
 setPin();                       //เรียกใช้งานฟังก์ชัน setpin()
}                                //จบฟังก์ชัน setup()
void loop() {                    //ฟังก์ชัน loop() สำหรับสั่งให้ทำงานที่ต้องการ
 turnOn();                       //เรียกใช้งานฟังก์ชัน turnOn()
 turnOff();                      //เรียกใช้งานฟังก์ชัน turnOff()
}                                //จบฟังก์ชัน loop()
//Fucntions
void setPin(){                   //สร้างฟังก์ชัน setPin()
  for (int indexPin = 0; indexPin < pinCount; indexPin++) {
                                 //คำสั่ง for วนลูปการกำหนดหมายเลขขาทั้ง 4 Pins
  pinMode(pinLed[indexPin], OUTPUT);
                                 //ฟังก์ชัน pinMode() ตั้งค่าทั้ง 4 Pins ให้เป็นโหมด OUTPUT
 }                               //ออกจาก for loop
 Serial.begin(9600);
     //ฟังก์ชัน Serial.begin สำหรับเปิดใช้งาน Serial Monitor ด้วย Baud Rate 9600
}                                //จบฟังก์ชัน setPin()
void turnOn(){                   //สร้างฟังก์ชัน turnOn()
 for (int indexPin = 0; indexPin < pinCount; indexPin++) {     //คำสั่ง for วนลูปทีละขา Pin 2, 3, 8, 13
  digitalWrite(pinLed[indexPin], HIGH);    //สั่งให้ไฟติดทั้ง 4 LEDs
 }                               //ออกจาก for loop
 Serial.println("Turn ON");      //แสดงข้อความและขึ้นบรรทัดใหม่
 delay(1000);                    //หน่วงเวลา 1 วินาที
}                                //จบฟังก์ชัน turnOn()
void turnOff(){                  //สร้างฟังก์ชัน turnOff()
 for (int indexPin = 0; indexPin < pinCount; indexPin++) {     //คำสั่ง for วนลูปทีละขา Pin 2, 3, 8, 13
  digitalWrite(pinLed[indexPin], LOW);     //สั่งให้ไฟดับทั้ง 4 LEDs
 }                               //ออกจาก for loop
 Serial.println("Turn OFF");     //แสดงข้อความและขึ้นบรรทัดใหม่
 delay(1000);                    //หน่วงเวลา 1 วินาที
}                                //จบฟังก์ชัน turnOff()
```

Step 5 : วิธีการทดลอง Arduino Tutorial 18

Tinkercad



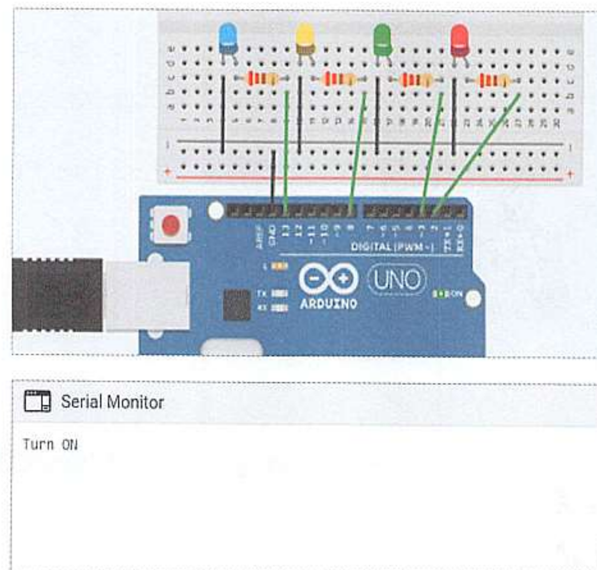
Arduino Board



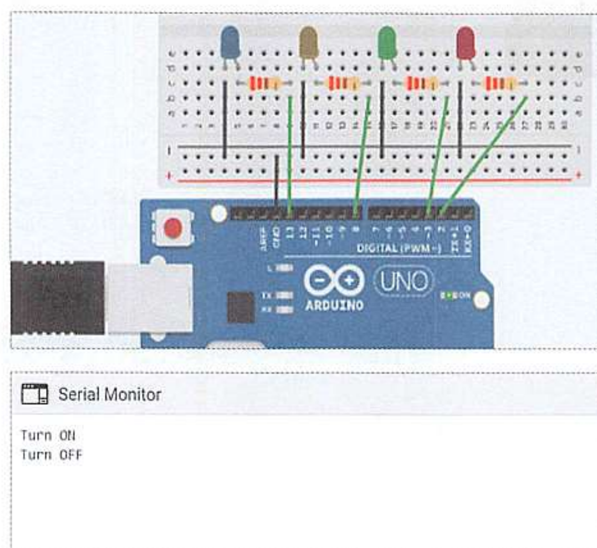
Step 6 : ผลการทดลอง Arduino Tutorial 18

Tinkercad Output

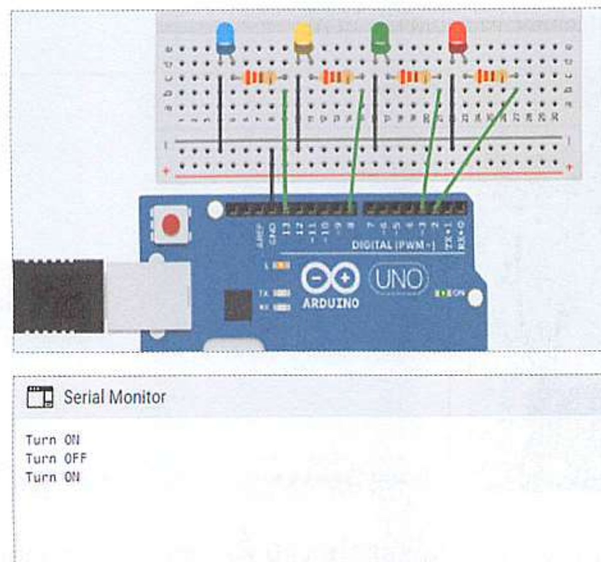
1. เมื่อรันผลโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติด พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



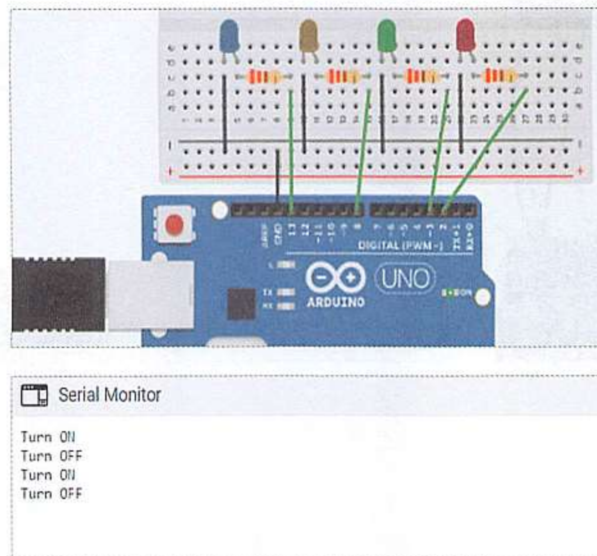
2. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับ พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที



3. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติดอีกครั้ง พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที

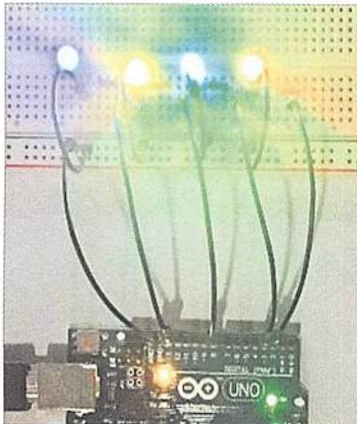


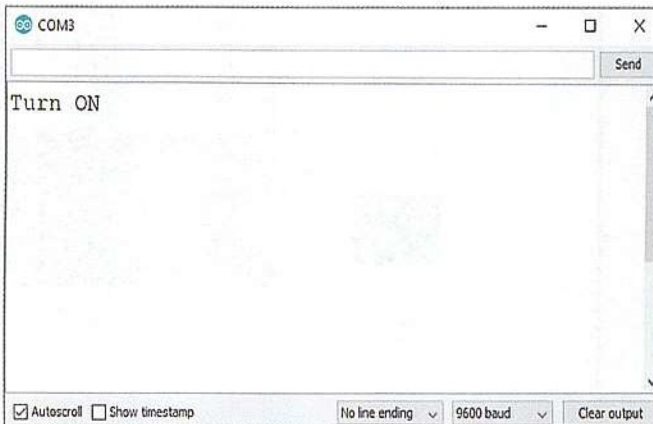
4. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับอีกครั้ง พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที และจะรันผลแบบนี้ไปเรื่อยๆ



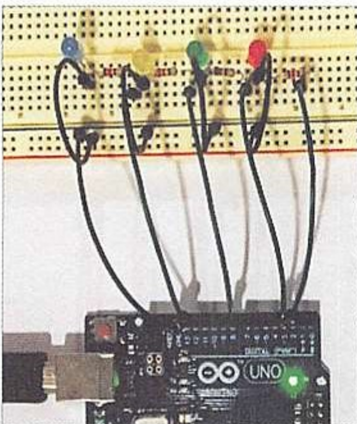
Arduino Board Output

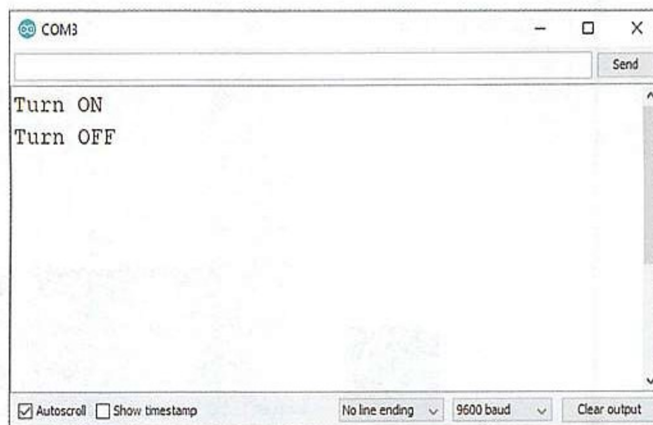
1. เมื่อรันผลโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติด พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



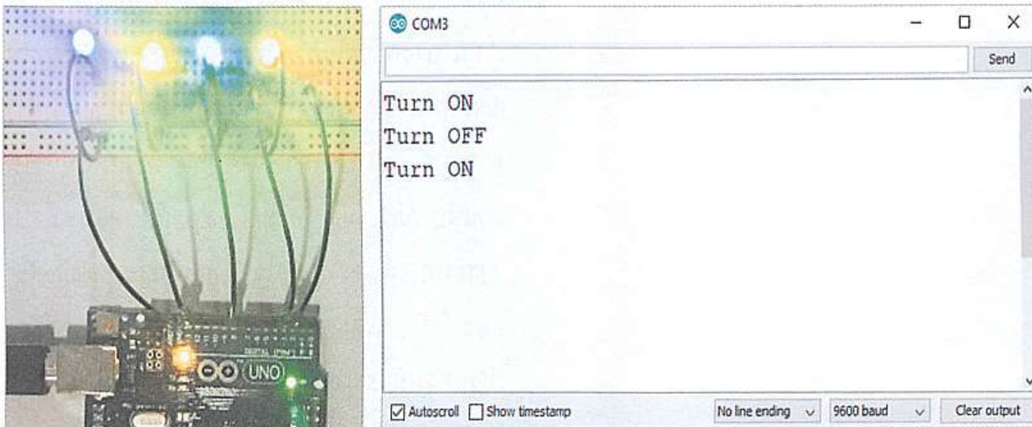


2. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับ พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที

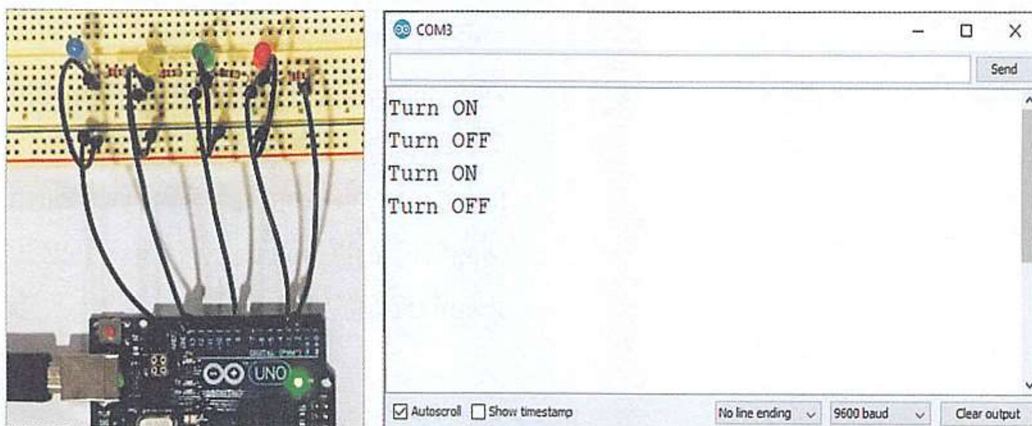




3. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงติดอีกครั้ง พร้อมแสดงข้อความ Turn ON บน Serial Monitor หน่วงเวลา 1 วินาที



4. หลังจากนั้นโปรแกรมจะสั่งให้หลอดไฟ LED ทั้ง 4 ดวงดับอีกครั้ง พร้อมแสดงข้อความ Turn OFF บน Serial Monitor หน่วงเวลา 1 วินาที และจะรันผลแบบนี้ไปเรื่อยๆ



Step 7 : คำถามท้าย Arduino Tutorial 18

1. ใน Arduino Tutorial 18 มีฟังก์ชันที่สร้างขึ้นเองกี่ฟังก์ชัน อะไรบ้าง
2. ฟังก์ชันที่สร้างขึ้นเองแต่ละฟังก์ชันทำหน้าที่อะไร
3. ข้อดีของการเขียนฟังก์ชันขึ้นมาใช้งานคืออะไร
4. ถ้าต้องการสร้างฟังก์ชันโดยให้แสดงข้อความ I love Arduino Programming หลังจากการกะพริบไฟในทุกๆ ครั้ง จะเขียนโปรแกรมอย่างไร

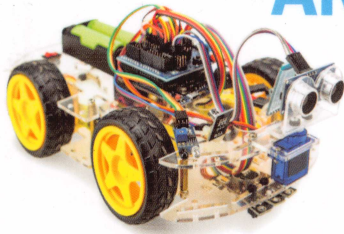
บทสรุปท้ายบท

การเรียนการเขียนโปรแกรมในบทนี้ เป็นการเรียนโครงสร้างไวยากรณ์ของภาษา C/C++ บนแพลตฟอร์ม Arduino ซึ่งมีลักษณะเฉพาะที่แตกต่างจากการเขียนโปรแกรมโดยปกติ ด้วยเหตุนี้จึงได้นำแล็บ Arduino Tutorial เข้ามาเป็นส่วนหนึ่งในการเรียน เพื่อจะได้เข้าใจ Syntax และ Parameter ของคำสั่ง และการนำไปใช้จริงในการเขียนโปรแกรมควบคุมการทำงาน

การสอนภาษา C อาจจะไม่ครบถ้วนครอบคลุมทุกคำสั่ง หรืออาจไม่ได้มีตัวอย่างการเขียนโปรแกรมที่หลากหลาย เหตุเพราะมีเนื้อหาอีกมากซึ่งต้องอาศัยเวลาอีกพอสมควรในการรวบรวมเรียบเรียงนั่นเอง เนื้อหาในบทนี้จึงถือเป็นจุดเริ่มต้นที่ไม่ยากเกินไปสำหรับมือใหม่หัดเขียนโปรแกรมควบคุมบอร์ดไมโครคอนโทรลเลอร์ ที่ต้องการก้าวแรกเพื่อสร้างกำลังใจ ก่อนจะมุ่งหน้าสู่การเรียนรู้ในระดับที่สูงขึ้นต่อไป

Practical Microcontroller Programming with ARDUINO

เรียนง่ายเป็นเร็ว



Arduino คืออะไร

คือ บอร์ดไมโครคอนโทรลเลอร์ที่ใช้สร้างโปรเจ็กต์ หรือสิ่งประดิษฐ์ที่เกี่ยวข้องในเชิงอิเล็กทรอนิกส์ วงจรไฟฟ้า และการเขียนโปรแกรม โดยมีชิปไมโครคอนโทรลเลอร์ควบคุมการทำงาน มีความหมายครอบคลุมทั้งฮาร์ดแวร์และซอฟต์แวร์ บอร์ด Arduino เป็นแพลตฟอร์มแบบ Open Source ที่เรียนรู้ง่ายไม่ซับซ้อน จึงได้รับความนิยมใช้พัฒนาโครงงานได้หลากหลาย เช่น หุ่นยนต์ของเล่น ระบบฝังตัว และระบบอัตโนมัติ

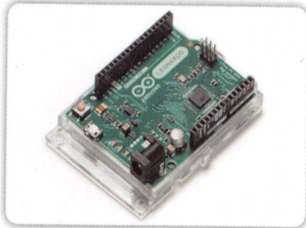
สร้างประสบการณ์ครบจบในเล่มเดียว

- สร้างทักษะพื้นฐานทาง Electronic, Circuit และ Programming
- เรียนเขียนโปรแกรมบน Arduino IDE ทั้งแบบ Online/Offline
- เรียนรู้วงจรทั้งแบบ Simulation ควบคู่กับการต่อวงจรจริง
- Book + VDO Online เพื่อการเรียนรู้ที่รวดเร็วและมีประสิทธิภาพ
- แถม Source Code แล็บการทดลอง (Arduino Tutorial Lab)
- สอดแทรก Link เชื่อมโยงไปยัง Tutorial/Datasheet/Document เพื่อต่อยอดความรู้ได้ทันที

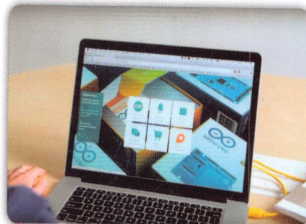
พัฒนาโปรเจ็กต์บน Arduino Platform ดีอย่างไร

- Arduino for Everyone ออกแบบมาให้ง่ายสำหรับทุกคน
- เป็น Open Source ทั้งฮาร์ดแวร์และซอฟต์แวร์
- มีแพลตฟอร์มออนไลน์ช่วยในการพัฒนา Project
- มีแหล่งชุมชนที่ใช้แลกเปลี่ยนความรู้ซึ่งกันและกัน
- มี Simulation Software ช่วยให้การเรียนรู้ง่ายขึ้น
- Arduino Products ครอบคลุมทุกระดับการศึกษา ทั้งนักเรียน นักศึกษา นักวิจัย และผู้เริ่มต้นที่ไม่มีพื้นฐาน

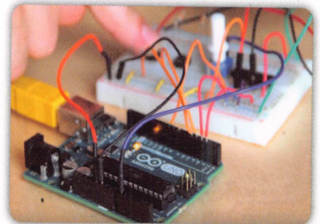
เรียนง่ายเป็นเร็วแบบ Super Speed



1 เรียนพื้นฐาน Microcontroller จาก Arduino Products

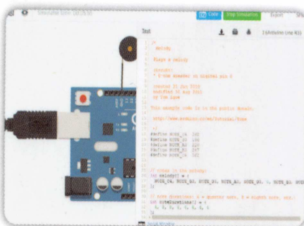


2 เข้าใจการพัฒนา Project บน Arduino Platform



3 ฝึกการต่อวงจร > โค้ดดิ้ง > โหลดโปรแกรมผ่าน LAB

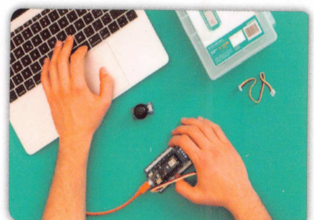
4 ทดสอบการทำงานได้ทันทีผ่าน Simulator Software



5 เห็นภาพชัด! เพราะมี VDO Clip ประกอบการเรียน



6 Arduino Programming ฉบับคนไม่เคยเขียนโปรแกรม





ผู้แต่ง ทศพล บ้านคลองสี่



เจ้าของเพจ/ยูทูบ : Aj. NesT The Series และ GlurGeek.com

บรรณาธิการ ภีรพล คชาเจริญ



ISBN 978-616-487-315-5

